

JAVA ITECH US

Some features include Object Oriented, Platform Independent, Multi-threaded

DAY 1

Introduction

- **JDK 1.6** is used in companies.... **JRE**- run java program **JDK** - develop java app
- 90% of companies them use **eclipse IDE** **Java Runtime Environment** is an implementation of the Java Virtual Machine which executes Java programs. It provides the minimum requirements for executing a Java application
- **JDBC** mysql/oracle
- **Servers** tomcat jetty weblogic **JIT compiler:** It improves the runtime performance of computer programs based on bytecode.
- Java program generates a **.class** which runs on **JRE**.
- Compiled version of the java app store in **bin folder** as a .class file
- Character strings are represented by the **class JAVA.lang.String**

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html>

Java language:

- **char set** (A-Z,0-9,+,-,/,*)
- **words (identifier)** it can be also known as variables...
eg: amount,rate,test_project
- **JAVA_HOME** = jdk location till bin folder

Variable:
`int x; // field declared to be of a primitive data type x
= 100; // initialization`

Local Variable: Variables defined inside methods, constructors or blocks are called local variables.

Instance variables: are variables within a class but outside any method. These variables are instantiated when the class is loaded.

Class Variable: These are variables declared with in a class, outside any method, with the static keyword

Java naming conventions (very imp for company)

- **Package**, or class-which can be helpful in understanding the code.
 - **com.google.maps.geocoding** grouping of related types (classes, interfaces, enumerations and annotations)
- (com folder - google folder -...)
 - lowercase with company details
- **Classes / Interfaces** - first letter of each internal word capitalized. - **Apple**

Method signature -

1. Method name
 2. Method PARAM's (no.of.params, and datatypes)
 3. Access modifier/specifiers
 4. Return type
 5. Exceptions it throws
- **Methods:** in mixed case with the first letter lowercase, with the first letter of each internal word capitalized. **A method is an ordinary member function of a class. It has its own name, a return type (which may be void), and is invoked using the dot operator.**
 - **Variables** - Variable names should not start with underscore _ or dollar sign \$ characters, even though both are allowed.
 - **float myWidth;**
 - **Constants** - The names of variables declared class constants and of ANSI constants should be all uppercase with words separated by underscores ("_").
`static final int MAX_WIDTH = 999;`

Core Java usually refers to Java SE which consists of the Java Language, the JVM and JDK. Core Java is used for general purpose programming and almost anything written in Java is based on the Core Java.

Enterprise Java refers to Java applications written for enterprises; the leading technology here is Java EE such JSP-----

access modifiers : public, private, default --> same package, protected

non-access modifiers : final , static, abstract

The **access specifier** determines how accessible the field is to code in other classes. You can optionally declare a field with an access specifier keyword: public, private, or protected.

DAY 2 Control Flow Statements

Multiple main methods

SWITCH - BREAK function....

public static void main(String[] args) {

 int month =4;

 String monthString;

 switch (month) {

 case 1: monthString = "January"; System.out.println(monthString); break;

 case 2: monthString = "April"; System.out.println(monthString); break;

 case 3: monthString = "June"; System.out.println(monthString); break;

 case 4: monthString = "October"; System.out.println(monthString);

 case 5: monthString = "November"; System.out.println(monthString);

 case 6: monthString = "December"; System.out.println(monthString);

}

Output:

October November December

```
int a=0;
while(a<10)
{
    if(a==5)
        break; // exits here
    System.out.println(a);
    a++;
}
```

```
for(int i=0; i<100; i++)
{
    if(i%2==0)
        continue; // skips even and prints odd numbers
    System.out.println(i);
}
```

- **break** - stop or exits out of the loop
- **continue** - skip a iteration/ move to next step
- Expressions are the core components of **statements**.
- Statements may be grouped into **blocks**.
- The most basic control flow statement supported by the Java programming language is the **if-then** statement.
- The **switch** statement allows for any number of possible execution paths.

For bitwise operator

<https://www.youtube.com/watch?v=JGtgBnSQEyg>

<http://www.java2s.com/Code/Java/Language-Basics/CatalogLanguage-Basics.htm>

Declaration - defining a variable to be of a specific datatype/class type

Initialization - assigning a value to the variable/field/property

Instantiation - calling new operator to create a new instance of that class

How objects are stored?

In java, each object when created gets a memory space from a heap.

DAY 3 Classes and objects

Class inside another class?

Yes we can write, but cannot declare it as public otherwise it will show error Only one class must have access modifier.

// Use "f" you have a decimal float number.

Class is a blue print from which individual objects are created. A class can contain fields and methods to describe the behavior of an object.

float f = 4;

So you need a float f = 4.5f;

double a = 2.45f;

Class - design your object

OOPS --- conversion of real time object into a java app

Object - stores some data and do some functionality.

Constructor - Class needs a constructor to construct an object.

Constructor gets invoked when a new object is created. Every class has a constructor. If we do not explicitly write a constructor for a class the java compiler builds a default constructor for that class.

Data hiding using **private** keyword.

If I declare variables **private** in sub class.... You can give access to **main** class by **getters and setters**.

Constructors vs methods:

Constructors must have the same name as the class and can not return a value.

They are only called once while regular methods could be called many times.

Super class: Fruits object = new Fruits();

System.out.println(object.getApple());

Sub class: private int apple = 500;

public int getApple() {

return apple;

}

public void setApple(int apple) {

this.apple = apple;

}

Instance:

variables are instantiated when the class is loaded. 'only' when you use 'new' operator and call the constructor to create a new instance of that class.

g = new String("abc"); // heap - instantiation

You are creating a new instance of 'String' class and initializing 'g' with it.

An **interface** is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.

Difference between class, Abstract class and instance-class?

Class → implements all of its methods □ can have instances □ can have sub classes (extends)

static/main/constructor

Abstract class → implements some, none or all of its methods □ cannot have instances □ must have sub class.

abstract keyword

Interface → implements none of its methods □ cannot have instances □ cannot have sub class but may have a class using implements

only abstract methods

supports multiple inheritance

interface keyword

Difference between Overloading and Overriding polymorphism satisfies both

Overloading → same method but different method signature □ □within the same class □ □return types may vary □ □Exceptions may vary

Overriding → same method and method signature but body may vary □ □execute in sub class □ □return types are same □ □Exceptions may same. □ □Void run () return 5 in super class □ □void run () return 10 in sub class.

Parameter – Variables assigned inside the methods and constructors.

Argument – Variables assigned inside an array such String [] args.

extends - to create a sub/child class

implements - provide implementations using method body for methods of an interface

YES :

class extends class → class implements interface → interface extends interface

NO :

class implements class → interface implements interface → interface extends class

DAY 4 Objects - OOPS concepts

java.lang.Object java API doc the base class in Java from which all classes are derived

Object class - class Car - Class Nissan extends Car....

Object class has a **toString()** method

Pen greenPen = new Pen(); System.out.println("Color via toString method: "+greenPen);

Output: Color via toString method: com.itech.corejava.Pen@7852e922

It will check for **toString()** method in main class if not it will display object class **toString()** method.

== And equals() process: **equals()** method is used to compare the contents of two string objects and returns true if the two have same value while **==** operator compares the references of two string objects.

String object == is working....

String object .equals is working....

Pen object == is not working....

Pen object .equals is not working....

String str = "Rama";

JVM - When Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by virtual Machine (JVM) on whichever platform it is being run.

String str2 = "Rama";

JVM will not create one more object as str2 reacts as duplicate of str1 as it has same data

In order to overcome...

String str2 = new String("Rama"); // creates new string object...

Inheritance: Check com.itech.oops.version1 and versionINHERIT in JAVA workspace
the process where a class acquires the properties of other class.

In order to **override method**.... Right click Source - Click Override which is used to change something different in sub class from the original in main class

@Override

In case of **Constructor in inheritance**, when a new object of a derived class is created, first the constructor of the super class is invoked and then the constructor of the derived class is invoked.

public void start() { System.out.println("BMW starts in a separate way"); }

- **Use instanceof** keyword can be used to test if an object is of a specified type. Inorder to test the new functionality changes which is created above “Overridden” one.

if(carRef instanceof BMW) { BMW bm = (BMW) carRef;

bm.moonRoof(); }

```
public class Father {  
int Fsalary = 50000;  
  
public class Son extends Father{  
int Ssalary = 30000;  
public static void main(String[] args) {  
Son S = new Son();  
System.out.println(S.Ssalary);  
System.out.println(S.Fsalary);
```

```
class Simple1{  
public static void main(String args[]){  
Simple1 s=new Simple1();  
System.out.println(s instanceof Simple);}//true
```

DAY 5 Objects- Inherit-Abstract

Inheritance:

```
Car carRef = new Honda();
```

carRef = new Nissan(); carRef object is a reference to Honda and Nissan objects in inheritance....

Object declaration in Inheritance hierarchy:

```
Object ob = new BMW(); // object Class -- Class Car -- Class BMW
```

```
ob.start(); // ERROR
```

Car c = (Car) ob; // type cast the Car object in order to call the Child class functionality from parent class

```
c.start(); // Now execute....
```

toString() method declaration in Inheritance:

```
Object obj = new Honda();
```

```
obj.toString(); // displays toString() method in Class Honda
```

```
Car car = (Car) obj;
```

```
obj.toString(); // displays toString() method of Class Honda
```

car.backcamera(); // ERROR: because the parent class reference object doesn't execute the child class functionality.

ABSTRACTION: It refers to the ability to make a class abstract in OOP. It helps to reduce the complexity and also improves the maintainability of the system.

```
public abstract class Car // Class must also be abstract because it contains abstract method.
```

```
{ public abstract void move(); } // because this method is not implemented here
```

- Implement this abstract method in the child class.... if not then declare the child class as a abstract. These classes cannot be instantiated. This class contains one or more abstract methods which are simply method declarations without a body.

Template design pattern: Used in Inheritance, if the method is not available in child class but executes the method which is available in parent class.

```
Parent class: start() {} stop() {}
```

```
executeCar()
```

```
{ start(); move(); stop(); }
```

Child class: move() {} The main benefit of encapsulation is the ability to modify our implemented code without breaking the code. With this Encapsulation gives maintainability, flexibility and extensibility to our code

Encapsulation: Wrapping of data and methods in the class. It is the technique of making the fields in a class private and providing access to the fields via public methods. If a field is declared private, it cannot be accessed by anyone outside the class, thereby hiding the fields within the class. Therefore encapsulation is also referred to as data hiding

Polymorphism: the ability of an object to take on many forms.

Two types of polymorphism -- compile time polymorphism and run time polymorphism.

```
public class Yamaha extends Bike{  
    public void start()  
    System.out.println("Bike has 40m/hr");  
    public static void main(String[] args)  
    Bike B = new Yamaha();  
    B.start();
```

```
public class Bike {  
    public void start()  
    System.out.println("Bike has 10m/hr");
```

DAY 6 Static/Final/Immutable Ob.

Static block: It is used to initialize the static data member, It is executed before main method at the time of class loading

Static or class level variables

```
static int count; // declared in parent class Employee.java
```

// declared in main class

```
System.out.println("Employee.count : "+Employee.count); // "static or class variable"
```

- Class followed by a variable "count" is static or class variable. Without an object.

```
emp1.count = 5; System.out.println("Emp1 object Count : "+emp1.count);
```

- Variable or a method declared with object is called instance variable or non-static variable or a instance method

```
emp2.count = 20; System.out.println("Employee.count : "+Employee.count);
```

```
System.out.println("Emp3 object without Count declaration: "+emp3.count);
```

Static Methods: // you can also declare static in methods

- Only static data is allowed and non-static or instance data is not accessible

```
// Parent class public static int add (int a, int b) { return (a+b); }
```

```
public static void sub(int a, int b){ System.out.println("Static VOID method SUB() : "+(a-b)); }
```

```
public void multiply(int a, int b) {
```

```
System.out.println("NON-Static VOID method Multiply() : "+(a*b)); }
```

```
// Main class System.out.println("Static INT method ADD() : "+CalculateUtils.add(5, 6));
```

```
CalculateUtils.sub(6,5);
```

```
CalculateUtils CU = new CalculateUtils(); CU.multiply(5, 5);
```

final keyword : (Constant) 1. variable 2. method 3. class

final in method: if I declare method as final in parent class, we cannot override in child class

final in class: if I declare class as final in parent class, we cannot extend in child class

Mutable Objects: Changing the state or value of an object after the creation of the object is called mutable objects emp.setSalary(5000);

Immutable Objects: No way of changing the state of an object.

String is an immutable object. StringBuffer is a mutable object. StringBuffer class is thread-safe uses append, insert, replace()
StringBuffer SB = new StringBuffer("Yunus");

String s1 = "Rama"; later I declare s1="Krishna"; It creates a new value with different reference.

In order to convert a mutable javaApp to immutable class String class overrides the equals() method of Object class.

StringBuffer class doesn't override

1. Make class a final

String is slow and consumes more memory when you concat too many strings because every time it creates new instance.

StringBuffer is fast and consumes less memory

2. Delete default constructor, setters and getters
StringBuilder is non-synchronized i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.

StringBuilder is more efficient than StringBuffer.

DAY 7 JAVA API

JAR files is Java Archive files and it aggregates many files into one. It holds Java classes in a library. JAR files are built on ZIP file format and have .jar file extension.

JAVA API - libraries - predefined resources

JRE - package of JVM + library (such as java.lang.System class or java.io)

Package into .jar File

System.out.println()

- System class - out variable from printStream class - println() method

<http://docs.oracle.com/javase/7/docs/api/> Click System

Comments Shortcut in Eclipse – Ctrl+Shift+/\

JAVA API library JAR file - Click Project - import - jar file...

Use those jar library files in other project and click "ADD built path"

```
import com.itech.oops.Static.Employee;
```

```
public class TestLibApp { public static void main(String[] args) { Employee emp = new Employee(); } }
```

Private Constructor: restricts the creation of an object outside the class

throws: If a method does not handle a checked exception, the method must declare it using the throws keyword. The throws keyword appears at the end of a method's signature.

```
Class ForName: public static void main(String[] args) throws Exception { executeCarLogic("com.itech.oops.versionClassFORNAME.BMW"); }
```

```
public static void executeCarLogic(String className) throws Exception
```

```
{ Class classObj = Class.forName(className); //Load your class into memory JVM }
```

```
Object obj = classObj.newInstance(); //Creates a new instance of the class represented by this Class object.
```

- If you use newInstance() - Your class must contain zero argument constructor.

```
if(obj instanceof Car) { Car carRef = (Car) obj; }
```

Shortcut for searching a class file or JAVA program: CTRL + SHIFT + T

Debugger: In order to get the argument passed. Right click - inspect

Program arguments: Setting arguments to (String args[]) - Click Debug configurations - type in arguments tab with a space

Environment variables: Every project has a production environment or Test environment

Click Debug configurations - arguments tab - VM arguments

```
-DCountry=USA -Dname=Yunus -Dlang=Eng //No spaces
```

```
System.out.println("Country : "+System.getProperty("Country")); // access via main method
```

FILE INPUT STREAM:

```
FileInputStream FISobject = new FileInputStream("G:\\JAVA-J2EE Workspace\\Java Project\\src\\com\\itech\\oops\\Static\\Employee.java");
```

```
int data; while((data = FISobject.read()) != -1) //Reads till end line in the File
```

```
{ System.out.print((char) data); FISobject.close(); } // type casting the data and prints the data
```

Variables : can store only one value int x = 10;

Arrays - store multiple values - it stores only one data type and problem is the size.

DAY 8 Collections - LIST

The List and Set extends the Collection interface. Should not confuse the Collection interface with the Collections class which is a utility class.

Collections: In other languages we call data structures.
Stack - Queue - LinkedList - Array - BinaryTree -
Sorting/Search algorithms

Collection (I)

List - is an interface which Allow duplicate elements which implements ArrayList, Vector, LinkedList, Stack maintain the order of values

Vector:
Synchronized
(2 different threads accessing at same time)
* Doesn't allow null

Vector class:
List listObj = new ArrayList(); provides the capability to implement //String
listObj.add("Yunus"); a grow able // int
listObj.add(800); array of objects // Class
listObj.add(new Employee()); // object
listObj.add(new Pen()); //object

// Read with the index

ArrayList: System.out.println(listObj.get(0));

Unsynchronized
Allows null

//Size

System.out.println(listObj.size());

//Delete

System.out.println("Before Deleting: "+listObj);

// Display all the contents

listObj.remove(2); // Remove using index
listObj.remove("Yunus"); //Direct object

System.out.println("After Deleting: "+listObj);
Java Generic methods and generic classes
with a single method declaration, with a single class declaration

Generics:

List<String> strList = new ArrayList<String>();
this list is generated for string objects, this concept is called Generics

List<Employee> empList = new
ArrayList<Employee>(); //Employee class

Iterator: Reads the elements and delete something, instead of using enhanced for loop (Read only)

Iterator<String> strIterator = strList.iterator();
while(strIterator.hasNext())

// You can use hasPrevious() for backward direction

```
{  
    String str = strIterator.next();  
    if(str.equals("Yunus"))  
    {  
        strIterator.remove();  
    }  
}
```

//deletes the items

```
{  
}
```

ListIterator: //Remove on both direction

ListIterator<String> strListIterator = strList.listIterator();

Array list:

- * uses dynamic array
- * act as a list
- * Slow access
- * better for storing and accessing

Linked list

- * doubly linked list
- * act as a list and queue both because it implements List and Deque interfaces
- * fast access
- * better for manipulating data.

Contains() in List:

```
if(strList.contains("Yunus"))  
// checks for the element in the list  
System.out.println("Element Found");  
else  
    System.out.println("Element not found");
```

```
List<String> searchList = new ArrayList<String>();
```

```
searchList.add("shreya");  
searchList.add("Yunus");  
searchList.add("faraaz");  
if(strList.containsAll(searchList))  
// checks for all elements with other list  
System.out.println("Search Elements found");
```

Contains() with object in List:

Student.java

```
@Override  
public boolean equals(Object obj)  
{  
    System.out.println("toString method  
"+this.toString()+" VS "+obj.toString());  
    if(obj instanceof Student)  
    {  
        Student stu = (Student) obj;  
        if(this.sNo == stu.getsNo() &&  
        this.sName.equals(stu.getsName()) && this.sFee ==  
        stu.getsFee())  
            return true;  
    }  
    return false;  
}
```

ContainsObjectProgram.java:

```
List<Student> stuList = new ArrayList<Student>();
```

```
stuList.add(new Student(100,"Yunus",8520));  
stuList.add(new Student(101,"zak",8522520));  
stuList.add(new Student(102,"irs",85250));  
stuList.add(new Student(103,"far",22520));
```

```
if(stuList.contains(new Student(100,"Yunus",8520)))
```

// It shows not found if you don't boolean in

Student.java

```
System.out.println("Student record found");  
else  
    System.out.println("Student record not found");
```

Set - Doesn't allow duplicate elements (Unique objects)

Map - Key value pair (SSN is key and name is value)

Thread scheduler in java is the part of the JVM that decides which thread should run. Under **preemptive scheduling**, the highest priority task executes until it enters the waiting or dead states or a higher priority task comes into existence. Under **time slicing**, a task executes for a predefined slice of time and then reenters the pool of ready tasks. The scheduler then determines which task should execute next, based on priority and other factors.

DAY 9 Collections - SET- SORT

Multithreading in java is a process of executing multiple threads simultaneously. But we use multithreading than multiprocessing because threads share a common memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

Collection is an interface, **Collections** is a class.

Sorting with list elements:

```
Collections.sort(listObj);
```

Sorting with list objects:

Main class:

```
Collections.sort(stuList);
```

//ERROR: as the list contains objects

- **SOLVED:** After implementing Comparable sorting logic in parent program. **The synchronization is mainly used to:**

```
System.out.println(stuList);
```

To prevent thread interference.
To prevent consistency problem.

Parent class:

```
public class Student implements Comparable<Student>
@Override // Custom sorting logic
public int compareTo(Student stu) {
    if(this.sNo == stu.getsNo())
        return 0;
    else if(this.sNo > stu.getsNo())
        return 1;
    else
        return -1;
}
```

Sorting with list objects Using Comparator:

Main Class:

```
SNameComparatorProgram SCom = new
SNameComparatorProgram();
Collections.sort(stuList,SCom);
//List and Sorting mechanism
```

Implementation Parent class from Student.java:

```
public class SNameComparatorProgram
implements Comparator<Student> {
    // Creates a sorting logic using name which
    implements Student class
@Override
public int compare(Student stu1, Student stu2) {
    return stu1.getsName().compareTo(stu2.getsName());
} // Compare Strings
```

Sorting with list objects Using Comparator with Anonymous class without creating an additional class:

```
Collections.sort(stuList,new Comparator<Student>()
{ public int compare(Student stu1, Student stu2) {
    return stu1.getsName().compareTo(stu2.getsName());
        // Compare Strings
}});
```

LinkedList: It is efficient in insertion and deletion of elements with references than ArrayList.

- Designed for nth or position based operations.

Vector: It is thread safe "multiple users trying to accessing your resource/object, then I need thread safe", ArrayList is not.

- It does not allow others to access, if this thread is currently accessing the object.

Trans: Set of statements executed on a database or file system

ThreadSafe mechanism used in money withdrawal from ATM and Online transaction at same time. **method can be accessed by only one thread at a time.**

Interview: Have you used Vector?

- Tell NO, I have been using ArrayList.

The life cycle of a thread

Newborn state

Runnable state

Running state

Blocked state

Dead state

Set - TreeSet, HashSet, LinkedHashSet

Main Class:

```
Set<String> strSet = new HashSet<String>();
```

```
Set<String> strSet = new TreeSet<String>();
```

- **TreeSet** - Eliminate the duplicate elements and sort the order.

```
strSet.add("Yunus"); strSet.add("Irshad");
```

```
strSet.add("Yunus"); strSet.add("Faraaz");
```

System.out.println(strSet.size()); //automatically delete duplicate elements

```
System.out.println(strSet);
```

Set for objects: System.out.println(stuSet.size());

// It

displays 8 because they are objects

Hashcode - helps in encryption...

Str "Yunus" = 6668 //reference

Str "Yunus" = 6668 // same because we use hashCode() object.hashCode();

// memory location with hashcode number

Interview: Why I need to overwrite a hashCode() or equals() method?

@Override

```
public int hashCode() {
```

```
    int code = sNo+sName.hashCode();
```

// two objects are equal then hashCode must be equal

- If two hashcodes are equal then it may or may not the objects are equal

```
System.out.println("Code: "+code);
```

```
    return super.hashCode(); }
```

Main class:

```
stuSet.add(new HashCodeStudent(100,"Yunus",8520));
//hashSet is gonna calculate hashCode
```

//duplicates

```
stuSet.add(new HashCodeStudent(100,"Yunus",8520));
```

Output:

Code: 85788468

Code: 85788468

Code: 120457

Code: 120457

TreeSet: Use HashSet always if the project require sorting mechanism this use TreeSet

```
Set<Student> stuSet = new TreeSet<Student>();
```

// ERROR: Student class need to have implemented method with comparable<Student> and compareTo() method.

anonymous class is a class defined without any name in a single line of code using new keyword.

```
public java.util.Enumeration testMethod()
```

```
    return new java.util.Enumeration()
```

3 constants defined in Thread class:
public static int MIN_PRIORITY 1
public static int NORM_PRIORITY 5
public static int MAX_PRIORITY 10

HASHSET: uses hash table to store the elements. It extends AbstractSet class and implements Set interface. * contains unique elements only.
LINKEDHASHSET: contains unique elements only like HashSet. It extends HashSet class and implements Set interface. * maintains insertion order.

DAY 10 Collections - MAP

Map - Whenever we pass the key and we gonna get the value...

```
Map<String, String> mapData = new  
Hashtable<String, String>(); // key can also be integer
```

```
Map<String, String> mapData = new  
HashMap<String, String>(); //you can use HashMap
```

synchronized

- HashTable is thread safe will not allow null values
- HashMap is NOT. will allow null values [
mapData.put(null,"King");]

```
mapData.put("101", "Yunus");  
mapData.put("102", "Faraaz");  
System.out.println(mapData.get("100")); //null
```

```
Set<String> keySet = mapData.keySet(); //Set of keys  
for(String key : keySet)  
    System.out.println("Key: "+key+"Value:  
"+mapData.get(key));  
    // Ordered is not guaranteed in Collections.
```

```
mapData.put("101", "YUNUS"); // It will  
override the element if the key 101 has "YUNUS".
```

```
System.out.println("Key is available:  
"+mapData.containsKey("101"));  
    // If key is available print TRUE
```

```
System.out.println("Value is available:  
"+mapData.containsValue("Faraaz"));  
    // If Value is available print TRUE  
http://docs.oracle.com/javase/7/docs/api/java/util/package-summary.html
```

Properties - `java.util.properties`

- extending the capabilities of hashtable
- Changing the languages in webpages like English German

database.properties file:

```
dbHostName=google.com./db  
userName=john  
password=pass123  
error1=Invalid User
```

- Properties accept only string....

```
Properties dbProp = new Properties();  
FileInputStream FIS = new  
FileInputStream("G:\\JAVA-J2EE Workspace\\Java  
Project\\src\\com\\itech\\properties\\databaseProp.prope  
rties");  
dbProp.load(FIS);  
System.out.println(dbProp); //display all the contents  
in properties file  
System.out.println(dbProp.get("dbHostName"));  
    // use the key to display specific data
```

Another Method: Recommended File reader technique
dbProp.load(PropertiesProgram.class.getResourceAsStream("databaseProp.properties"));

Third party Collection API:

- It is not part of java code "external lib" like apache commons.

Multi Valued map: One key but many values. (101, Ravi),(101, Yunus),(101, Zak)

JAVA APP contains IOUtils apache commons IO
2.4/collections and CollectionUtils

- ADD in classPath

FileReader from Other from IO and Utils:

ReadFileUsingUtils

```
{  
    String file = FileUtils.readFileToString(new  
File("G:\\JAVA-J2EE Workspace\\Java  
Project\\src\\com\\itech\\properties\\databaseProp.prope  
rties"));  
    System.out.println(file);  
    // Highly Recommended technique  
    List<String> list =  
IOUtils.readLines(ReadFileUsingUtilsProgram.class.ge  
tResourceAsStream("databaseProp.properties"));  
    for(String print : list)  
        System.out.println(print);  
}
```

Real time project environment:

- Develop
- Testing/QA
- User Acceptance Testing
- Stage
- Production like "well fargo"

Serialization is used when data needs to be transmitted over the network. Using serialization, object's state is saved and converted into byte stream .The byte stream is transferred over the network and the object is recreated at destination

DAY 11 CollectionUtils/Serialization

CollectionUtils from apache commons

1. Bidirectional Map-

```
BidiMap<String, String> bm = new TreeBidiMap<String, String>();
bm.put("100", "Yunus");
bm.put("103", "Zakira");
System.out.println("The Value is :" +bm.get("103"));
//if you know the key, you will get value
System.out.println("The key is :" +bm.getKey("Yunus"));
// if you know the value, you will get the key
bm.remove("103"); // remove the value by key
bm.removeValue("Yunus"); // remove the key by value
```

2. Multivalued map - allowing it to have more than one value for a key.

```
MultiValueMap<String, String> mv = new MultiValueMap<String, String>();
mv.put("USA", "AZ"); mv.put("USA", "FL");
mv.put("USA", "CA"); // Value: [AZ, FL, CA]
System.out.println("Value: "+mv.get("USA"));
```

3. Search - PredicateSearchExample:

```
List<Student> stuList = new ArrayList<Student>();
stuList.add(new Student(100,"Yunus",8520));
stuList.add(new Student(101,"zak",8522520));
List<Student> searchList = new ArrayList<Student>();
for(Student stu : stuList) //Student object
{
if(stu.getsFee() > 20000 && stu.getsName().startsWith("f"))
{ searchList.add(stu); }
}
System.out.println("Search List: "+searchList);
Another method: java2s predicate chain- useful in combining multiple conditions


- Predicate minimizes the for loop execution.
- Download Java beanUtils lib from apache commons


EqualPredicate eq = new EqualPredicate(8520);
// Fee value
BeanPredicate bp = new BeanPredicate("sFee", eq);
//Fee property and predicate
//ERROR: so download Collections 3.2 version lib from apache commons
List<Student> searchList = (List<Student>)
CollectionUtils.select(stuList, bp); //Collections "StuList" and condition "Predicate"
//ERROR: so we need to download logging lib from apache commons
```

MAVEN build tools maintain these libraries instead of downloading and uploading in class path.

- There are diff predicate equal, greater, lesser etc.

Marker/Tagged interface -

Interview: Have you implemented any marker/tagged interfaces?

Yes, interface with no methods

Cache - temporary memory area, which is used to store and fetch data.

eg: country list, in app, there are many registration pages..

We gonna use this cache mechanism to store country list.

```
public interface LuxuryCar { //no methods }
```

BMW implements Cache,LuxuryCar

Honda implements Cache

Nissan implements LuxuryCar

```
for(Object obj : list)
```

```
{ if(obj instanceof LuxuryCar)
```

```
{System.out.println("Yes it is a Luxury interface: "+obj);}
```

```
} else if(obj instanceof Cache)
```

```
// eliminate else... BMW will print both interfaces
```

```
{System.out.println("Yes it is a Cache interface: "+obj)}
```

```
System.out.println("General interface..." +obj);
```

```
//Every object will generate this logic
```

- Generally we don't use marker interfaces in real time apps.

Interface cloneable - copy of an object.....

Student stu = new student(101,"yunusd"2121);

Student stuclose = (Student) stu.clone();

//ERROR: mark the student to interface clonable.

Serialization: sharing information from JVM1 to JVM2 (remote or different machine) via network.

- Conversion of java object into stream "writing".
- Convert the java file into XML/JSON file.
- Only java can read this serialization process
- JVM2 will read the file by deserialization method.

Interview: Why we need a serialization?

**No one will write this logic in real time...

public class Light implements Serializable{

```
// all your fields must be serializable
```

```
private transient Student student;
```

// student class not to be serializable we use transient

SerializationApp:

```
FileOutputStream FOS = new FileOutputStream("LightFile.ser");
```

//creates Output File

```
ObjectOutputStream OOS = new ObjectOutputStream(FOS);
```

```
//writes the file object
```

```
OOS.writeObject(l); // write the light object
```

DeserializationApp:

```
FileInputStream FIS = new FileInputStream("LightFile.ser");
```

//reads serialization File

```
ObjectInputStream OIS = new ObjectInputStream(FIS); //reads the file object
```

```
Light l = (Light) OIS.readObject(); // reads the light object
```

Extensible: custom serialization...

private static final long serialVersionUID:

- Object has an identification, ID never change of Light object.
- We did a de-serializable by SVID, de-serializable will allow if not it won't allow.... checks same object.
- Serialization not used in enterprise

Why we need serialize?

Google.com - distribute their system to accommodate the load across the world. They use small algorithm **LOAD BALANCER**: how many cluster you have and which is free & busy?

GMAIL1 server / cluster - if fails GMAIL2 server / cluster

GMAIL3 server / cluster - if busy, then it directs to server 1 // called session efficiency

- We can have multiple catch block?
 * Catch the null pointer or any other first....
 * Then catch all the exceptions like Exception e or catch (Throwable E).....

An error is an irrecoverable condition occurring at runtime. Such as OutOfMemory error.

DAY 12 EXCEPTION HANDLING

exception is a problem that arises during the execution of a program.

Interview: Can we have a class without a main method?

- Yes, but any one of the class in the APP must contain a main method in order to run the APP. It complies but doesn't run.

Exception Handling:

Interview: How do you handle Exception in your project?

If there is a one-step mistake, we have to handle using the fallback logic otherwise it will stop all steps in java app.

```
try{           takephoto();
}catch(Exception e){           It is not necessary that each try block
{                           must be followed by a catch block.
   // show error to user
}
```

- try{ //single try block so if one fails then it will fail full system
 - String s = null; // NullPointerException
 - e.printStackTrace();
- //print what exception/ which statement causes the error...

Interview: What is checked and unchecked exception?
 ERROR: forcing you to handle exception.

```
public static void main(String[] args) throws Exception {           // Checked exception
  • while executing main method, we are expecting an
    exception "IO" or "FileNotFoundException"
```

```
String months[] = {"Jan","Feb","Mar"};
System.out.println("Month is "+months[4]);
  • this part may cause Exception but not prompting so
    UNCHECKED exception

  • Any Exception implements or extends
    RunTimeException then it is called UNCHECKED
    exception.
```

Exception hierarchy:

```
Object –
  throwable –
    Exception – ABCExcep (Checked exception)
      RunTimeException –
        ABCException
          (unchecked exception)

two main subclasses :
  * IOException class
  * RuntimeException Class.
```

CustomException:

```
public class CustomException extends Exception{
  private String errCode;
  private String errMessage;
  public CustomException(String errMessage)           // shows an error message to user
  {
    super(errMessage);
    //passing my error message to super class
  }
  public CustomException(String errCode, String errMessage)
  {
    super(errMessage);
    this.errCode = errCode;
    this.errorMessage = errorMessage;
  }
```

CustomExceptionApp:

```
public class CustomExceptionApp {
  public static void main(String[] args) throws CustomException {
    int result = divide(10,0);
    // Warning: this method may cause
    Exception so throws
    System.out.println(result);
  }
  public static int divide(int a, int b) throws CustomException           //this method needs a throws to handle
  {
    if(b ==0)
    // Exception occurred as b cannot be zero
    {
      CustomException CE = new CustomException("Error205","B cannot be
zero...");
      throw CE;
      // inform to caller then throw it....
    }
    return a/b;  }
```

DAY 13 EXCEPTION - MAVEN

used in exception handling for it will execute even if exception is not occurred

Real time Exception handling mechanism:

- APP divided into multiple layers or classes.

Layers depend of APP architecture

- UI layer** - bunch of presentation logic - will react on UI exceptions with error messages
- Business layer** - business logic - bunch up the exceptions.
- Service layer** - may cause 10 more service Exceptions.
- Data layer** - may cause 10 Exceptions.

User enter a username and password.... request is handle by UI layer - pass through layers

- Handles all exceptions and finally comes with a common exception.

Refer Java Programs:

```
com.itech.Exceptionhandling
```

```
CustomException
```

```
    DataException
```

```
        BusinessException
```

```
            DataLayer
```

```
                BusinessLayer
```

```
                    ExceptionLayerApp
```

- In company, you may get many unchecked exceptions like ArrayBound etc.

Interview: How do you handle exception in your application?

- We have multiple layers in our application, so each layer may cause different exception.
- We design one custom exception "Data Exception" - handle all 10 Exp with Error code and Error msg - create a new Data exception(100,"DB down");

Interview: How to handle a null pointer exception?

- Most of the time we check null pointer exception...In order to avoid write conditions like if(String data!=null)
- In rare case we create some CustomException

```
int n = 7;
```

```
int result = 1;
```

```
for (int i = 1; i <= n; i++)
```

```
    result = result * i;
```

FACTORIAL PROGRAM -->

```
System.out.println("The factorial of 7 is " + result);
```

finally: try{ //success block

```
    catch { //failure block
```

```
    finally {
```

// work in both case success or fail

Interview: when you will use finally....

- Closing a resources like file, DB connection etc.

Diff between final, finalize, finally?

finalize method: which is executed before when the object is removed from the memory.

Maven build tool: 98% in companies use this.

packaging APP

- Instead of downloading libraries and adding to class path.
- Ant - outdated version.... Gradle - Latest version.
- Automate your lib management from public repository.

Private repo - client specified library system.

Click New Maven project - simple project - Group

ID/Artifact: MavenApp

Configuration file: pom.xml

// here we are adding lib information

Search for maven configurations -

<http://mvnrepository.com/>

<dependencies> <!-- Copy the dependency after searching -->

```
<dependency>
```

```
<groupId>commons-collections</groupId>
```

```
<artifactId>commons-collections</artifactId>
```

```
<version>3.2.1</version>
```

```
</dependency> You can use tools like Ant, CruiseControl,
```

```
</dependencies> and Maven etc to continuously build
```

and integrate your code.

Interview: Garbage collection in JVM architecture?

- Just tell I don't know.... It uses garbage collection to free the memory. By cleaning those objects that is no longer reference by any of the program.

Interview: Discuss Out of memory exception?

- can be solved in performance test... when they test 1000 customers in one minute

FIBONACCI PROG -->

```
int n1=0; int n2=1;
```

```
for(int i=1; i<10; i++)
```

```
    int n3=n1+n2;
```

```
    System.out.println(n3);
```

```
    n1=n2;
```

```
    n2=n3;
```



```

public class Single {
    private static Single s;
    private Single() {
    }
    public static Single giveMeInstance() {
        if (null == s) {
            s = new Single(); // create
        }
        return s;
    }
}

```

Transaction manager:

In Hibernate transaction, you need to use `HibernateTransactionManager`. If you only deal with pure JDBC, use `DataSourceTransactionManager`; while JTA, use `JtaTransactionManager`.

DAY 15 JDBC - SQL INJECTION

Read/SELECT database from JAVA APP:

```

String SQL = "SELECT * FROM student";
ResultSet rs = stmt.executeQuery(SQL);
                                //retrieve the elements row
while(rs.next())
{
    // iterating to each row with column name.
    System.out.println("Sno: "+rs.getInt("sno"));
}
                                // best read the value with column name
rs.close(); con.close();
// these closing resources takes place under finally block

```

Interview: If there is a mistake in SQL statement, who knows it JAVA APP or database?

* JAVA APP doesn't know, but passes the SQL command to database for verification and execute.

SQL Injection: (Hacking a APP will harm your data)

- Instead of passing a data via webpage input, we are passing SQL command.
- Hackers can get all data by this SQL injection.

Passing parameters in SQL:

```
String SQL = ("SELECT * FROM student WHERE fee > "+fee+" AND sname= '"+name+"'");
```

- In program arguments in Run configurations: SQL injections.

"(SELECT fee FROM student WHERE sname='hameed')"

faraaz

- They can delete without knowing your APP:

"(DELETE fee FROM student WHERE sname='hameed')"

faraaz

- A smart hacker might get access to all the user names and passwords in a database by simply inserting 105 or 1=1 into the input box.

"(SELECT fee FROM student WHERE sname='hameed') OR
1=1"

Interview: Diff between Prepared Statement and Statement?

Statement - Parse/read - validate - execute statements for 1000 records

PreparedStatement – pre compiled SQL statement and execute only once, this avoid the SQL injection.

PreparedStatement: READ

```

String SQL = ("SELECT * FROM student WHERE sname = ? ");
PreparedStatement pstmt = con.prepareStatement(SQL);
pstmt.setString(1, name);
ResultSet rs = pstmt.executeQuery();

```

INSERT:

```

public static void insertParameters(int sno, String name, String course, int fee) throws Exception
{
    String SQL = ("INSERT INTO student VALUES (?, ?, ?, ?)");
    PreparedStatement pstmt = con.prepareStatement(SQL);
    pstmt.setInt(1, sno);
                                //array index changes in INSERTING
    pstmt.setString(2, name);
    pstmt.setString(3, course);
    pstmt.setInt(4, fee);
}

```

INSERT 1000 Records:

```

for(int i = 1000; i<2000; i++)           // pre complied
{
    pstmt.setInt(1, i);
    pstmt.setString(2, i+"Raj");
    pstmt.setString(3, i+"BBA");
    pstmt.setInt(4, i+1000);
    pstmt.executeUpdate();
}

```

Transaction management is required to ensure the data integrity and consistency in database.

Transaction management:

- Used in INSERT/UPDATE/DELETE operations
- If anyone Statement STMT fails --> No change or update anything in DB

Ex: transfer funds from source to destination, if it is success, then debit from SRC and credit in DEST.

If it fails, then do nothing....

`con.setAutoCommit(false);`

```

try{                               //don't save the records automatically
    Statement stmt = con.createStatement();
    stmt.executeUpdate("INSERT INTO STUDENT values
    (1,'FIRST','MS',10000)"); //stores in DB
    stmt.executeUpdate("INSERT INTO STUDENT values
    (2,'SECOND','MS',20000)"); //stores in DB
    stmt.executeUpdate("INSERT INTO STUDENT values
    (1,'FIRST','MS',10000)"); //Fails: duplicate record
// Stores in some memory location.....
    con.commit();                // success and save into database
} catch(Exception e)
{
    con.rollback();              // fails and rollback "UNDO" the transaction
}

```

Batch Updates:

- Sending all information in one statement as a group instead of multiple statements (insert,delete,update)
 - Helps in improving the performance of the application.
- ```

stmt.addBatch("INSERT INTO STUDENT values
(4,'FIRST','MS',10000)");
stmt.addBatch("INSERT INTO STUDENT values
(3,'SECOND','MS',20000)");
stmt.addBatch("INSERT INTO STUDENT values
(5,'FIRST','MS',10000)");
stmt.executeBatch(); .insertBatchSQL(sql);
 // grouping the batch

```

Use `update()` when you are sure that object already exists and is associated to a session

`.insertBatch(customers);`

Use `merge()` when you are not sure that object exists and is associated to a session

**Stored Procedure**

May or may not return a value  
Called by EXEC command  
Can call function

**Function**

Must return a value  
Called from SQL statements  
Cannot call procedure

**Trigger**

Code fragment that runs before or after a row or table is modified

# DAY 16 JDBC – DAO/DTO/DBCP

**Callable statement:**

- Used to call the stored procedures
- Using PL/SQL (apart from tables such as functions/procedures) in the database.

**Interview:** You use stored procedures and how you call it?

- I am not familiar with PL/SQL but I know to call stored procedures.
- I can trigger those functions using callable statement by passing parameters.

**Interview:** Diff between Statement/PreparedStatement/CallableStatement?

In MySQL workbench, create stored procedures. insert one record in student DB

```
CallableStatement call = con.prepareCall("call addStudent()"); // calling the procedure
```

**written stored procedures in PL/SQL:**

```
CREATE PROCEDURE insertStudent (sno int, sname varchar(20), course varchar(20), fee int, out result varchar(50))
BEGIN
 INSERT INTO student VALUES(sno,sname,course,fee);
 SET result = "Student inserted successfully";
END
```

```
CallableStatement call = con.prepareCall("call insertStudent(?, ?, ?, ?, ?)");
call.setInt(1, 27);
call.setString(2, "Alia");
call.setString(3, "MSC");
call.setInt(4, 54353);
call.registerOutParameter(5, Types.VARCHAR);
// Outputs
call.executeUpdate();
String result = call.getString(5);
// Student inserted successfully
System.out.println(result);
```

**DAO : Database access objects**

- Concept is same as DBUtil.getDBconnection();
- Instead of writing bunch of SQL commands in all classes, we write DB related queries in single studentDAO class.

**Spring DAO** uses `org.springframework.jdbc` package

to provide all the JDBC related support required by your application

**DTO - Data transfer object**

- Holding your transfer data may be input or output using serializable interface.

StudentDTO implements Serializable

StudentDAO

StudentDAOApp

**StudentDAO:** // getting the records from database and storing in object and return it

StudentDTO stuDTO = new

```
StudentDTO(rs.getInt("sno"),
rs.getString("sname"), rs.getString("course"),
rs.getInt("fee"));
lis.add(stuDTO);
```

**StudentDAOApp:** //Get all student details

```
List<StudentDTO> listDTO = sdao.getAllStudents();
for(StudentDTO dto : listDTO)
 System.out.println(dto);
```

**DB Connection management (Connection pooling mechanism):**

Creating and closing the connection takes some time....

- Third party lib:** C3po (Statement pooling), Apache DBCP in pom.xml

**4 parts - driver - URL - username - password**

**DataSource** --> interfaces (server, DBCP, C3PO) maintain the database connections.

```
private static Properties prop = new Properties();
private static BasicDataSource BDS = new BasicDataSource();
static // pre create the database connection
{ try {
prop.load(DBUtil.class.
getResourceAsStream("database.properties"));
BDS.setUrl(prop.getProperty("dbURL"));
BDS.setUsername(prop.getProperty("userName"));
BDS.setPassword(prop.getProperty("password"));
BDS.setDriverClassName
(prop.getProperty("className"));
BDS.setInitialSize(30); // 30 connections are allowed.
```

```
public static Connection getDBconnection() throws
Exception
{ return BDS.getConnection(); }
```

- Real time uses Data source/server maintains 100 connections (delete idle connections)

**Core:** BeanFactory, IOC engine

**Context:** Config file to define enterprise services (beans.xml)

Two ways --> XML configuration or Annotation based

# DAY 17 SPRINGS - Introduction

when system grows large in Java project, the huge object dependencies will always tightly coupled causing objects very hard to manage or modify. In this you can use Spring framework to act as a central module to manage all the object dependencies easily and efficiently.

**Framework** – will follow all standards and makes the development faster.

The **basic Idea of Spring Framework** is

- You do not create an object, but describe how they should be created, by defining in Spring Config. File
- You do not call a services and components, but tell which services and components must be called, by defining in spring configuration files
- Spring is going to handle dependency objects.
- Dependency object hierarchy:

StudentApp → StudentDAO

- DBUtil
- BasicDataSource
- DBProperties

Hibernate works with **Plain Old Java Objects (POJOs)**, which is much like a JavaBean

**Interview:** Advantages of Spring framework:

- **Layered architecture**: Reduces time on configuration, creation/maintenance and spring takes care

- **Open source (free)**: Application classes are self-documenting, and dependencies are explicit.  
- **IOC, DI and AOP**: It provides an easily usable view for application code  
Works on POJOs Plain Old Java Objects (POJOs) programming. Hence easier for dependency injection / injection of test data. enables reuse component.  
Flexible use of Dependency injection. Can be configured by XML based schema or annotation-based style.

**Dependency injection: define the dependency of components on services, so that those services are wired.**

**Several types of IOC Dependency Injection:**

- Setter, Constructor, Method or Interface Injection.

**Interview:** What are features of Spring?

Lightweight, Inversion of control (IOC), Aspect oriented (AOP), Container, MVC Framework, Transaction Mgmt

**Interview:** What does the Spring framework do?

- Spring is a framework that helps you to "wire" different components together.
- Dependency injection promotes very easy unit testing
- It also provides declarative transactions, job scheduling, authentication, and a bunch of other functionality

**Interview:** What JDK and Springs version you are using?

- Am using JDK 6 and sometimes I use JDK 7
- I used Springs 3.0 Version

**Convert Spring project into JDK environment**

- Right click java build path. Remove JRE and add JDK
- Spring uses Apache log4j framework lib which shows the step by step performance of spring's execution.

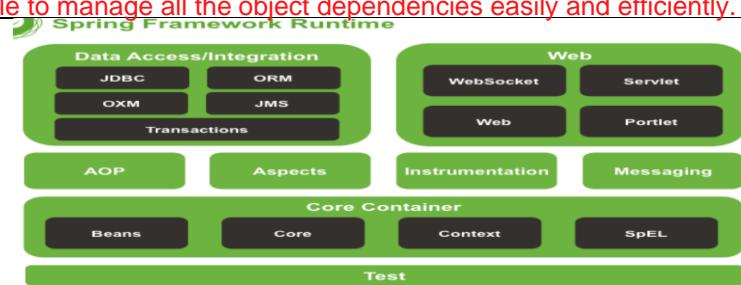
**ClassPathXmlApplicationContext** : It Loads context definition from an XML file located in the classpath, treating context definitions as classpath resources.

**FileSystemXmlApplicationContext** : It loads context definition from an XML file in the filesystem. The application context is loaded from the file system by using the code

**XmlWebApplicationContext** : It loads context definition from an XML file contained within a web application.

**Spring Life Cycle :** - Container finds bean definition from config file and instantiate the bean

- Using IOC dependency injection , populates all properties as specified in config file      - Call init()      - Call service methods



• Spring core module - (IOC - Inversion of control means configuring of bean "java objects") like core java  
**You do not create an object but define how it is to be created, so that they are automatically instantiated at runtime**

• Java bean- any independent object such as others like POJO, DTO(contains sets/getters) and Domain object

**Java beans** representing form inputs containing request parameters from the view, referencing the action bean.

**Declaring the version details in one place in pom.xml:**

```
<properties><org.springframework.version>4.1.6.RELEASE</org.springframework.version></properties>
<version>${org.springframework.version}</version>
```

**Spring 'auto-wiring'** make development faster with great costs  
**ApplicationContext.xml:**

```
• Configure student object as spring bean, Configuration driven object into XML in resources.no - default AW
<bean id="studentID" byName - prop. Name constr - byType in cons
class="com.itech.coresprings.Student"> autodetect - defau.cons
//Student studentID = new Student();
<property name="sno" value="81" />
//Injecting the student data
```

```
<property name="addr" ref="addressID" />
</bean> //Dependency injection: stu.setAddress(addr)
//Student studentID = new Student(82,"Jo","MS",3);
<bean id="studentID1" class="com.itech.corespri.Student"
autowire="byType"> Spring Auto wiring features to
// autowire must have just one copy of address object
// wire beans in other beans.
// The Spring container can autowire relationships
value. between collaborating beans.
// scan all spring all object such as address object
```

```
<constructor-arg name="sno" value="82" /> </bean>
<bean id="addressID" class="com.
itech.coresprings.Address">
<constructor-arg name="city" value="Miami"/> </bean>
```

**"p" schema injection:** inside bean p:name="yunus" />

**SpringApp:**

//Load Spring configuration file is called as Spring container (Reading the property file and get an object)

```
ApplicationContext AC = new
ClassPathXmlApplicationContext("ApplicationContext.xml");
//returning a spring object
```

Student stu = AC.getBean("studentID",Student.class);

**//Singleton / Prototype: object is a sharable object. in 20 locations**

//Only one object "Stu" below are references

```
Student stu1 = AC.getBean("studentID1",Student.class);
Student stu2 = AC.getBean("studentID",Student.class);
```

**Prototype:** Cloning of an object to reduce the cost of creation. E.g. method overriding

**Request:** Single bean definition maps to life cycle of a HttpRequest  
**Session:** Single bean definition maps to life cycle of a HttpSession  
Only valid in the context of a web-aware Spring ApplicationContext.

# DAY 18 SPRINGS

## Lazy initialization

- It is the tactic of delaying the creation of an object.

```
<bean id="ID" class="classname" lazy-init="true">
```

## Aggressive:

```
<bean id="ID" class="com.itech.coresprings.Student">
```

## Prototype: Single definition maps to multiple object instances

- It helps in creating brand new object every time.

```
<bean id="ID" class="Class name" scope="prototype">
```

Message : Message by custA Message : null

## Singleton -- scope default is singleton

- init() -- some logic Message : Message by custA
- destroy() -- callback or finally Message : Message by custA
- After object creation, execute init method and destroy method. It doesn't retrieve object content again.

```
<bean id="ID" class="class name" scope="singleton" init-method="init" destroy-method="destroy">
```

## Interview: Define Singleton design pattern?

- Creation of only one object, then class follows a singleton pattern.

## Only one instance of the object exists at anytime

```
private static payRollSingleton payobject = null; //only one object is created
private payRollSingleton() // Private constructor
 // will not allow to create an object
```

```
public static payRollSingleton getInstance()
 //helps in creating an object
if(payobject==null)
 payobject = new payRollSingleton();
return payobject;
```

```
public class SingletonApp {
 payRollSingleton pay = payRollSingleton.getInstance();
 //id=17
 payRollSingleton pay1 = payRollSingleton.getInstance();
 //id=17
```

## Interview: Spring follows a singleton pattern?

No, spring will maintain and share singleton object using scope.

## Springs in JDBC BasicDataSource:

```
DataSource DS =
AC.getBean("bdsID",BasicDataSource.class);
System.out.println(DS.getConnection());
<bean id="bdsID"
class="org.apache.commons.dbcp.BasicDataSource">
<property name="url"
value="jdbc:mysql://localhost:3306/maydb" />
```

In most scenarios,

you just need to make sure a particular property has been set, but not all properties..For this case, you need @Required annotation

A Customer object, apply @Required in setPerson() method to make sure the person property has been set.

Define custom @Required-style annotation in Spring: you will create a custom @Required-style annotation named @Mandatory

@Retention(RetentionPolicy.RUNTIME) @Target(ElementType.METHOD)

## HttpRequest scope

```
Mark mark1 = context.getBean("mark");
Mark mark2 = context.getBean("mark");
mark1 == mark2; //This will return true
```

## Prototype scope

```
mark1 == mark2; //This will return false
```

## Collections in Springs:

```
<bean id="colRef" class="class name">
<property name="stateList">
 <list> <value>FL</value> </list> </property>
<property name="dataSet"> <!-- It support any data -->
 <set><value>1</value>
 <ref bean = "studentIDref" />
<property name="stuMap">
 <map>
 <entry key="Key 1" value="1" />
 <entry key="1" > <entry key="Key 2" value-ref="PersonBean" />
 <bean class="com.itech.coresprings.Student" >
 <!-- Injecting the student data -->
 </entry>
<property name="dbprop">
 <props> <prop key="dbURL">localhost</prop>
<bean id="studentIDref"
class="com.itech.coresprings.Student">
<property name="sno" value="81" /> //Injecting the data
<property name="sname" value="John" />
```

## Interfaces in Springs framework:

```
<bean id="bmwID" class="springs.Interface.BMW" />
 <!-- BMW obj = new BMW() -->
<bean id="carAppID" class="springs.Interface.CarApp" >
<property name="car" ref="bmwID" />
 <!-- already bean is created for HONDA -->
```

```
public class CarApp {
 private Car car;
 // Using the interface and writing the logic
public void executeCar()
 car.start(); car.move(); car.stop();
```

```
ApplicationContext AC = new
ClassPathXmlApplicationContext("CarApplicationContext.xml");
CarApp CA = AC.getBean("carAppID",CarApp.class);
CA.executeCar();
```

**ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);**

## Annotation:

\* By using this you can bypass the XML configurations.

\* Used when there is no injection of data in XML file.

```
 @Component("carAppID")
 public class CarApp {
 @Autowired
 @Qualifier("nissanID")
 private Car car;
 }
```

- No need getters and setters...

## In XML:

```
<context:annotation-config/>
<context:component-scan base-
package="com.itech.springs.Annotations" />
```

**@Service  
@Scope("prototype")**

**public class CustomerService**

## types of annotation

### bean:

required \* pre destroy  
component \* auto scan

The 'ListFactoryBean' class is a way to create a concrete List collection class (ArrayList and LinkedList) in Spring's bean configuration file. It will instantiate an ArrayList at runtime, and inject it into a bean property.

```
<property name="lists"> <bean class="org.springframework.beans.factory.config.ListFactoryBean">
<property name="targetListClass"> <value>java.util.ArrayList</value> </property> <property name="sourceList"><list> <value>1</value>
```

# DAY 19 SPRINGS - JDBC

## Download Spring JDBC lib

Declare a **PropertyPlaceholderConfigurer** in bean configuration file and map to the 'database.properties' properties.

```
<property name="location">
<value>database.properties</value>
```

### Interview: About JDBC in Springs ?

We have used Class JDBCTemplate to perform complete operations.

In Spring, **InitializingBean** and **DisposableBean** are two marker interfaces,

\* IB --> it will run afterPropertiesSet() after all bean properties have been set.

\* DB --> it will run destroy() after Spring container is released the bean. context.close();

**JDBCTemplate:** in XML: init-method="initIt" destroy-method="cleanUp">

in Annotation: @PostConstruct @PreDestroy

- JDBCTemplate depend on the database object.

- We can avoid, Connection, STMT/PSTMT, RESULT SET etc.

**JdbcTemplate**, you save a lot of typing on the redundant codes, it will handles automatically.

**JdbcDaoSupport**, set the datasource and JdbcTemplate in your class is no longer

required, you just need to inject the correct datasource into JdbcCustomerDAO

```
JdbcTemplate JDBCtemp = AC.getBean("JDBCTemplateID",JdbcTemplate.class);
```

```
int count = JDBCtemp.queryForInt("SELECT COUNT(*) FROM demotable");
```

```
System.out.println(count);
```

**Spring inheritance** is supported in bean configuration. A child bean can inherit its parent bean configurations, properties and some attributes. We cannot declare a bean when abstract="true"

```
<bean id="CustomerBean" parent="BaseCustomerMalaysia">
```

```
<bean id="JDBCTemplateID" class ="org.springframework.jdbc.core.JdbcTemplate" >
```

```
<constructor-arg name="dataSource" ref ="bdsID" />
```

```
<bean id="bdsID" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
```

```
</bean><property name="driverClassName" value="com.mysql.jdbc.Driver" />
```

```
<property name="url" value="jdbc:mysql://localhost:3306/test" />
```

```
<property name="username" value="root" />
```

```
<property name="password" value="password" /> </bean>
```

## JDBCTemplate in SQL operations:

- Reduces complexity

```
private JDBCTemplate temp;
```

```
temp.update(SQL,args);
```

```
temp.insert(SQL,args);
```

\* SPRINGS IMPORT: Use annotation procedure with @import

```
public class App {
public static void main(String[] args) { //Normally, you will split a large
Spring XML bean files into multiple small files, group by module or category}
```

```
ApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);
```

```
@Configuration
@Import({ CustomerConfig.class, SchedulerConfig.class })
public class AppConfig {
}
```

### ALTERNATIVE:

```
<property name="lists">
<util:list list-class="
java.util.ArrayList">
<value>1</value>
```

'**SetFactoryBean**' class provides developer a way to create a concrete Set collection (HashSet and TreeSet) in Spring's bean configuration file.

```
<property name="sets">
<util:set set-class="java.util.HashSet"
> <value>1</value>
```

The '**MapFactoryBean**' class provides developer a way to create a concrete Map collection class (HashMap and TreeMap)

```
<property name="maps">
<util:map map-class="
java.util.HashMap">
<entry key="Key1" value="1" />
```

**In Spring framework**, whenever a bean is used for only one particular property, it's advise to declare it as an **inner bean**.

```
Customer [person=Person [address=ashburn, age=25,
name=Yunus]]
```

### How to load multiple Spring bean configuration file

```
* This will work but takes lot of time to execute.
ApplicationContext context = new
ClassPathXmlApplicationContext(new String[] {"SpringBeans.xml","Copy of SpringBeans.xml"});
```

\* Combine using "import" Spring bean configuration files into a single XML file.

### Bean accessing in different XML files or inside file

```
<import resource="Copy of SpringBeans.xml" />
<ref bean="someBean"/>
```

### Bean accessing in same XML file

```
<ref local="someBean"/>
```

# DAY 20 WEB APPLICATIONS

## Standalone applications:

- APP with a main method.

## Web applications:

### Client (browser)

- Request/Response - HTML/CSS/JS

### Server

- Process/Deliver (JAVA logic) - Servlets/JSP

## Role in Web APP:

- Web developer will design the page and give it as a template to JAVA developer then we have to integrate the functions.
- Click Developer tools in Google Chrome

### Interview: Are you comfortable in working with WebSphere?

- Yes, I know how to work with it. Web APP is generic so I can work with any server

### Interview: What version you used?

- Tomcat 6.0 / 7.0

## DNS Server:

- Contains HTTP number with a port. It is hard to remember HTTP number, So we named as Domain name in google.com:8080 (port number)
- You can buy domain name from Godaddy.com

## WebApp folder:

In that create a WEB-INF folder:

- Create web.xml (Web deployment descriptor and configuration file) must in WEB-INF

- In creating web APP from Maven click war as packaging tool.

This is Web Archive File and used to store XML, java classes, and JavaServer

**SMTP:** Simple Mail transfer protocol used for mailing purposes

## Deploy the APP into Server:

- Right server - Add/Remove - move APP

## Access HTML file via server:

- Executes HTML page in webapp folder  
http://localhost:8080/WebAppProject/Emp/Employe e.html

## HTTP status codes:

HTTP 2xx: Success

HTTP 3xx: Redirecting...

HTTP 4xx: Error

HTTP 5xx: Database Error Exception

## Web Standards HTTP methods:

### GET (default)

- Everything related a request to server.
- All information appended to URL  
GET?username=yunus&password=yunus
- We have limitations.

### POST - <form method="POST">

- Sharing information to server. will not append in URL
- We use POST in real time for user specific information

### PUT/DELETE

- Server will disable this operation. Because client can delete the file.

### Interview: Interview: Diff between GET and POST method?

### Action

- For which resource you want to send data  
<form action="http://www.ticketnew.com" />

**Interview: Once the code is moved to the production? Issue occurs, How you will solve?**

- Depending on the server log.

Protocol - set of rules that determine how data is transmitted.

Hypertext - Hypertext is text which contains links to other texts.

# DAY 21 WEB APP - SERVLETS

It generates dynamic content and interacts with client through Request and Response.

## Servlet:

- Java object which process the request and respond to it.
- Need Servlet and JSTL lib. Use Maven

## Servlet - Interface

GenericServlet - use for FTP/SMTP- init() - service()- destroy()

HttpServlet - use for HTTP init()- service()- destroy()- doGet()- doPost()  
Project Servlets....

```
public class ServletApp extends HttpServlet
{
@Override
public void service(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException
{
String uname = request.getParameter("username");
String pass = request.getParameter("password");

System.out.println("Username: "+uname+" and
Password: "+pass);
```

```
PrintWriter out = response.getWriter();

if(uname != null && uname.equals("yunus"))
{
 out.println("Welcome to Java class....");
}
else
{
 out.println("Login failed.....");
}
```

## Define your Servlet in terms of URL in web.xml:

```
<servlet>
 <servlet-name>ServletApp</servlet-name>
 <servlet-class>com.itech.servlet.ServletApp</servlet-
class>
</servlet>
<servlet-mapping>
 <servlet-name>ServletApp</servlet-name>
 <url-pattern>/ServletApp</url-pattern>
</servlet-mapping>

<welcome-file-list>
 <welcome-file>Login.html</welcome-file>
</welcome-file-list>
```

## Connecting Servlet action from HTML:

```
<form action="ServletApp" method="post">
```

## Passing your parameters:

```
<form action="https://login.yahoo.com/">
```

- You can pass the parameters with equal name both in yahoo.com and your code...

USERNAME: <input type="text" name="username" />

https://login.yahoo.com/?username=yunus&passwd=yun  
us

- But server rejects your username and password to enter
- But using POST you can redirect and enter with correct credentials.



Sign in to your account

yunusirshad

\*\*\*\*\*|

Keep me signed in

Sign in

## HTML code in Servlet program:

```
out.println("<html><body><h1>Welcome to Java
class....</h1></body></html>");
```

## Redirection:

- Redirecting to other page from Servlet program

```
response.sendRedirect("Emp/Employee.html");
```

control goes to UI and then gets re  
routed to other JSP with new request.

## Another method:

```
RequestDispatcher RD =
request.getRequestDispatcher("Emp/Employee.html");
RD.forward(request, response);
```

same request is passed to  
other jsp and control does  
not go to UI.

<jsp:include page = " " /> - response of the second jsp is included in the response of the first jsp

<jsp:forward page = " " /> - request is routed to several other jsp before response is sent back

<jsp:param name = " " value = " " /> - Parameters you pass along, while forwarding

**JSP Life Cycle:** When a request is mapped to a JSP page, the web container first checks whether the JSP page's servlet is older than the JSP page. If the servlet is older, the web container translates the JSP page into a servlet class and compiles the class. After the page has been translated and compiled, the JSP page's servlet (for the most part) follows the servlet life cycle.

# DAY 22 JSP SCOPE - SERVLETS

JSP stands for Java Server Pages. It is a presentation layer technology independent of platform. They are like HTML pages but with Java code pieces embedded in them.

**JSP** - UI logic + java objects from server side

Directives -- Declarations -- Expressions -- Scriptlets

**RequestDispatcher.include()**

- Combine the objects output from many resources.
- This directive tells the container to merge the content of other external files with the current JSP.

**Ex:** Header and footer in Web page not going to change in all web pages...only content area changes.  
just include page="header.jsp";

**Request Scope:**

- Once the request is done it expires.

```
<jsp: class="EmployeeBean" scope="request" />
```

**Sending output of one page as input to another page:**

```
request.setAttribute("key",value);
```

- Retrieve by request.getAttribute(key);

**Page Scope:** similar to local variable

**Session Scope:** session object is on server side until your session expires (logout in Gmail)

- Store user specific information

```
HttpSession.setAttribute("key",value);
```

**Application scope:**

- All pages until you restart the server.
- Store application data

```
ServletContext.setAttribute("key",value);
```

**Interview: How do you maintain your session?**

**Session(Login/logout):**

- HTTP is a stateless protocol- server is not going to maintain user conversation.
- To maintain user conversation we use Session object.
- If client USER shares its session details like ID to server...if matches then gives the session with all details.
- Session will be OPEN still you close the browser such as GMAIL.

**Client requested for an associated servlet**

- First look into WEB.XML
- Creates a servlet object & Execute service method

**Cookies**

- A piece of information stored on client machine (Session ID)
- Client can restrict those cookies, GMAIL will not work if you disable cookie.

**JSESSIONID:** seen in URL, they don't care about whether the cookie is enable/disable.

An object of ServletConfig is created by the web container for each servlet. Used to pass initialization parameters to the servlet

**Interview: Diff between ServletContext and ServletConfig?**

- Specific to one servlet      `<init-param>` tags are used
- Deploying an APP in server, it default maintains ServletContext.
  - ServletConfig object - meant for a specific page or resource and read configuration for this specific page. An object of ServletContext is created by the web container at time of deploying the project
  - Some configurations are global configurations which are read by Servlet Context. Applies to all servlets
  - Local is Servlet configuration. Used to define context
  - ServletConfig can be used under Application scope

parameters common to all `<context-param>` tags are used for servlets

**Life cycle of Servlets:** Call back methods

**init() method**

If an instance of the servlet does not exist, the web container

- Information to be executed in server only one time.
  - a. Loads the servlet class.
  - b. Creates an instance of the servlet class.

**service() method**

c. Initializes the servlet instance by calling the init method.

- Multiple requests make create multithreading environment
  - d. Invokes the service method, passing request and response objects.

**destroy() method**

- only one time it will execute...  
multithreaded program contains two or more parts that can run concurrently.

**Interview: can I call destroy method?**

Yes I can call manually, before deleting the object and delete it.

Thread can be created by:

\* implementing Runnable interface

\* extending the Thread class.

**Interview: Which one is output GET/POST/SERVICE?**

`doGet(req,res)`

```
System.out.println("GET");
```

`doPOST(req,res)`

```
System.out.println("POST");
```

`Service(req, res)`

```
System.out.println("SERVICE");
```

- Service method will always execute, if there is no Service method then GET or POST may print.

**Advantage of ServletConfig:** The core advantage of ServletConfig is that you don't need to edit the servlet file if information is modified from the web.xml file.

**Advantage of ServletContext:** Easy to maintain if any information is shared to all the servlet, it is better to make it available for all the servlet.

## Advice Action taken by Aspect at a Joint Point. Can be done in 2 ways :

1. Annotations based - it is defined by a 'point cut expression'. 2. Schema based – everything defined using <aop: tags in the config file

**Before advice** – Run before the method execution Ex: HijackBeforeMethod implements MethodBeforeAdvice

**After returning** – Run after the method returns a result **After throwing** – Run after the method throws an exception **ThrowsAdvice**

# DAY 23 SPRINGS - AOP

**After advice (finally)** – advice executes after joint point, whether exits normally or with exception

**Around advice** – Run around the method execution, combine all three advices above. (MOSTLY USED)

```
HijackAroundMethod : Before method hijacked!
Customer name : Yunus Basra
HijackAroundMethod : Before after hijacked!
HijackAroundMethod : Before method hijacked!
Customer website : http://www.yunus.com
HijackAroundMethod : Before after hijacked!
HijackAroundMethod : Throw exception hijacked!
```

You need Spring Aspects lib, CGLIB lib, SLF4J API

- Recent Real time projects are using this.

--> it's just an interceptor to intercept some processes, for example, when a method is execute, Spring AOP can hijack the executing method, and add extra functionality before or after the method execution.

### Aspect oriented programming:

- If the functionalities are same in all pages or layer
- For ex: if there is same method in 1000 java classes, if there is a change in method, so we declare in one java class so we use AOP.

Modularize cross cutting concerns such as logging, transaction mgmt etc.

### Interview: Why we need spring AOP?

- My service needs to call X and Y logic instead of executing my existing logic.
- If there is any update we need then I have to change in every place... so I can easily handle using Spring AOP
- If there is a JAVA class, we need to inject the business logic without touching the code so we can use Spring AOP
- We can avoid code tangling and code scattering.

### Code tangling:

- A function is doing some other functions, such as combining more functions together.
- Payment function instead of doing payment it execute other logic.

### Code scattering:

- We are spreading the same functionality or code everywhere in the APP.

### Interview: Where I have used AOP?

- Used in Audit logging, Transaction management, security check...

Spring's AOP technique is allow developers to manage the hibernate transaction declarative.

### AOP Steps:

In AOP you cannot inject your logic in between....

- You can add before or after existing logic or if execution takes place.

### 1. (ASPECT) what you want?

Audit/Security/Management

\* ASPECT - I like to inject some logic into the code dynamically.

### 2. (POINT CUT) Where you want to execute?

patterns (package/ class/ method level) Indicate which method should be intercept, by method name

3. When you want to execute? 2 ways: \* name match (before method or after or exception) \* REG EXP match

Joint Point: A certain point in the execution of a program

**AOP IN XML:** <bean id="hijackAroundMethodBean" class="com.mkyong.aop.HijackAroundMethod" />

<bean id="customerServiceProxy" class="org.springframework.aop.framework.ProxyFactoryBean">

<property name="target" ref="customerService" /> <property name="interceptorNames"> st></value>hijackAroundMethodBean</value>

```
public class EmployeeAOPApp
```

```
BasicConfigurator.configure();
```

```
//Inorder to avoid those log error add
```

```
BasicConfigurator
```

```
ApplicationContext AC = new
```

```
ClassPathXmlApplicationContext("AOPApplicationCon-
```

```
text.xml");
```

```
EmployeeService service =
```

```
AC.getBean("empServiceID",EmployeeService.class);
```

```
Employee EmpObj =
```

```
AC.getBean("empID",Employee.class);
```

```
List<Employee> emplist =
```

```
service.findAllEmp("Yunus"); .findById(1);
```

```
//Employee emp = EmpObj.findEmp("102");
```

```
EmpObj.setEno("2015");
```

```
//Employee is not a spring bean so this function will
```

```
not execute
```

```
System.out.println(emplist);
```

```
System.out.println(EmpObj);
```

```
@Aspect
```

```
public class LoggingAspect {
```

```
Logger log =
```

```
LoggerFactory.getLogger(LoggingAspect.class);
```

```
@After("execution(* find*(..))")
```

```
//Point CUT with find methods
```

```
public void writeLogStmt(JoinPoint JP)
```

```
//JP is to access all the parameters
```

```
{
```

```
//Aspect B.L
```

```
log.debug("Log Aspect... Invoking
```

```
method"+JP.getSignature().getName());
```

```
// display method name
```

```
Object[] arguments = JP.getArgs();
```

```
for(Object obj : arguments)
```

```
log.debug("Log Aspect arguments...."+obj);
```

```
// display parameters
```

**Advisor** – Group 'Advice' and 'Pointcut' into a single unit, and pass it to a proxy factory object. if you want all of your DAO classes in customer module to implement the AOP feature with SQL logging support (advise), then you have to create new proxy factory beans manually, as a result you just flood your bean configuration file with tons of proxy beans

```
<context:component-scan base- package="com.itech.springs.AOP" <!-- enable AOP -->
```

```
<aop:aspectj-autoproxy /> <bean id="LogAspectID" BeanNameAutoProxyCreator OR
```

```
DefaultAdvisorAutoProxyCreator class="com.itech.springs.AOP.LoggingAspect" />
```

```
<bean id="empID"
```

```
class="com.itech.springs.AOP.Employee" >
```

```
<property name="eno" value="321" />
```

```
<property name="salary" value="23123" /></bean>
```

# DAY 24 JUNIT – MOCK TEST

JUnit is used for Unit testing an open source framework.

- I like to test this add method without testing in main method.....
- Junit lib needed.... To run JUnit you should have JUnit.jar in your classpath.
- As we are writing test cases, we use test folder in maven project

## Test cases:

- Positive testing: passing positive numbers
- Negative testing: passing negative numbers...

```
public class Calculator {
 public int add(int a, int b)
 {
 return a+b;
 }

 public Double getInterest(int amount, int rate, int
months)
 {
 return new Double ((amount*rate*months)/100);
 }
}
```

## Calculator Test in test folder:

```
public class CalculatorTest {

 Calculator cal = new Calculator();
 @Test
 public void testAdd() // test the add method
 {
 Assert.assertEquals(11,cal.add(5, 6));
 // Expected result and the function
 }
 //@deprecated - try to avoid they may remove in next
version of JUnit
}
```

- Run as JUnit test

**Another method:** Avoid annotation style, we use public class CalculatorTest extends TestCase

```
@Before
public void init() - public void setUp()

//@After
public void destroy() - public void tearDown()
```

{ \*\*\* Create the test case

```
@param testName name of the test case/
public AppTest(String testName)
{ super(testName); }
```

```
/** * @return the suite of tests being tested */
public static Test suite()
{ return new TestSuite(AppTest.class); }
```

```
/** Rigorous Test :- */
public void testApp()
{ assertTrue(true); }
```

## MOCK Framework:

- If Service is not available then we use MOCK framework....
- In order to test the business logic we use MOCK framework....
- We are creating a dummy objects.
- JMock, EasyMock, PowerMock we use Mockito lib

## Interview: when do you use MOCK framework and its importance?

- My logic is using interface or service
- Service is not available, then mock creates an object for interface.
- If specifically anybody is calling this method then return the value.

```
public interface CalculateTaxInterface {
 int getIncomeTax(String state, int income);
 int getSalesTax(String state, int amount);
}
```

```
public class ShowTax {
 private CalculateTaxInterface CTI;
 public ShowTax(CalculateTaxInterface CTI)
 this.CTI = CTI;
```

```
public int printSalesTax(String state, int amount)
 return CTI.getSalesTax(state, amount);
public int printIncomeTax(String state, int income)
 return CTI.getIncomeTax(state, income);
}
```

```
public class ShowTaxTest {
 ShowTax ST;
 @Before
 public void init(){
 //Mock Frame is created interface object
 CalculateTaxInterface CT =
 Mockito.mock(CalculateTaxInterface.class);
 }
```

```
Mockito.when(CT.getSalesTax("FL",
2500)).thenReturn(1000);
Mockito.when(CT.getIncomeTax("FL",
2500)).thenReturn(1000);
ST = new ShowTax(CT);
}
```

```
@Test
public void testprintSalesTax()
{
 Assert.assertEquals(1000, ST.printSalesTax("FL", 2500));
 • Make import static org.mockito.Mockito.*; so that you
can avoid Mockito word in the code.
```



# DAY 26 MVCMModel - log4j - JSTL

In **Spring MVC**, **BeanNameUrlHandlerMapping** is the default handler mapping mechanism, which maps URL requests to the name of the beans. `<bean name="/welcome.htm" class="com.mkyong.common.controller.WelcomeController" />`

**ControllerClassNameHandlerMapping** use convention to map requested URL to Controller class\_<--> controllerName

**SPRING MultiActionController** is used to group related actions (add, update, delete, list) into a single controller.

**Enterprise JavaBeans(EJB).** A server-side component, which manages the architecture for constricting enterprise applications and managed. They are: **Session Beans:** Expanded as "Stateful", "Stateless" and "Singleton", A Remote or Local interface is used to access the EJB files. **Message Driven Beans (MDB):** Asynchronous execution by means of messaging paradigm is supported.

# DAY 27 JMS

Java Message Service (JMS) to provide the ability to act as a message consumer and perform asynchronous processing between the services.

## Java messaging service

- Contains interfaces and JMS server

## Application Server - extra functionalities

- WebApp server - FTP – SMTP – JMS - EJB
- weblogic, jboss, websphere.
- Tomcat/jetty is not an application server.

**MQ servers** such as ActiveMQ Apache, Microsoft MQ IBM MQ (popular one)

**Sync** - waiting for the response (Phone call)

**Async** - request and response (SMS)

- New Employee created notifications to team member.

## Interview: why have you used JMS?

### Advantages:

- implement an asynchronous communication (initiate the request and do not wait for response)
- Message delivery is guaranteed even Receiver is not available

- Sender - JMS server - Receiver(Listener)

- Sends msg - phone signal - receive msg

- JMS sends acknowledgement to sender that I accepted your message and sent to receiver.

## Sending configurations:

### 1. One to One (QUEUE)

### 2. One to many (TOPIC) - promotional offers to all vendors

## Real time example:

- Sender - JMS server - many receivers car rental 15\$/day (one too many) everybody will get message, if somebody has a car then they respond.
- ActiveMQ Apache JMS server acts as a Middle layer

## Interview: What version of ActiveMQ you used?

\* Version 5.5

## Activate JMS server ActiveMQ:

In command prompt, --> cd Apache active mq bin folder  
--> activemq --> Enter → Server Started....

Open the JMS Server <http://localhost:8161/admin>  
Queues tab

**Java developer role** - just send message and receive

## Interview: How to establish a connection in JMS?

ActiveMQ Lib and slf4j lib

- ConnectionFactory - tcp://localhost:61616
- Connection by name
- Queue/Topic
- Sender/receiver
- close the connection.

## Types of message sent:

- TextMsg - XML/JSON/ TEXT (99%)
- ObjMsg - not advisable
- MapMsg

## QueueSender:

//Get connection factory

```
ConnectionFactory CF = new ActiveMQConnectionFactory(ActiveMQConnection.DEF_BROKER_URL); //Get the connection
Connection con = CF.createConnection();
//start the connection con.start();
//Once connection started give me a connection.
Session ses = con.createSession(false, Session.AUTO_ACKNOWLEDGE);
// JMS gives an acknowledgment as we received your message
```

// Create a queue in Localhost queues

```
// Create the destination (Topic or Queue)
Destination destination = ses.createQueue("FirstQueue");
// Create a MessageProducer from the Session to the Topic or Queue
MessageProducer MP = ses.createProducer(destination);
for(int i=0; i<10; i++)
{
 TextMessage message =
 ses.createTextMessage("App message number : "+i);
 MP.send(message);
 System.out.println(message.getJMSMessageID());
}
```

// reply back some message ID

## QueueReceiver:

```
MessageConsumer MC
=ses.createConsumer(destination);
MC.setMessageListener(new QueueListener());
// Listens request every time....
Message message = MC.receive();
while(message !=null)
{
 if(message instanceof TextMessage)
 {
 TextMessage TM = (TextMessage) message;
 message = MC.receive();
 }
}
```

**SPRING Dependency checking modes:** feature to make sure the required properties have been set or injected.  
**4 dependency checking modes are supported:** \* **none** – No dependency checking. (default) \* **simple** – If any properties of primitive type (int, long, double...) and collection types (map, list..) have not been set, UnsatisfiedDependencyException will be thrown. **dependency-check="simple"**

# DAY 28 SPRINGS LOGIN-ERRORS

\* **objects** – If any properties of object type have not been set, UnsatisfiedDependencyException will be thrown. \* **all** – If any properties of any type have not been set, an UnsatisfiedDependencyException will be thrown.

**Define Errors in Login.jsp** as key in properties file.

Username.required=Username cannot be empty

- Key will be same in all other language properties file  
Username.required=Username\_DK cannot be empty

**In properties for label:** Username.Label=Username  
**In JSP:** <Springtag:message code="Username.Label" />

- If client machine is FRENCH then it fallback to english

```
<bean id="messageSource"
class="org.springframework.context.support.Reloadable
ResourceBundleMessageSource">
<property name="basenames" >
<list><value>classpath:errors</value>
 <!-- Configurations files -->
<value>classpath:labels</value>
</list></property></bean>
```

**Show ERROR messages in Login pages:**

```
<Springform:errors path="Username" cssClass="error"
/>
```

```
public class LoginValidator implements Validator {
public boolean supports(Class<?> clazz) {
 return User.class.isAssignableFrom(clazz);
}
public void validate(Object target, Errors errors) {
ValidationUtils.rejectIfEmptyOrWhitespace(errors,
"userName", "userName.required");

@Autowired
private LoginValidator loginValidator;
@RequestMapping(method=RequestMethod.POST,value="/login")
public ModelAndView validateLogin(User user,Errors errors){
ModelAndView mv = new ModelAndView();
loginValidator.validate(user, errors);
if(errors.hasErrors()){
 redirectToLogin(user, mv);
}}
```

User Name	<input type="text"/>	Username can't be blank/empty
Password	<input type="password"/>	Password can't be blank/empty
<input type="button" value="Login"/>		

**Hidden fields:** to maintain the user conversations.

- Client will not see this fields.
- Store each page result in terms of hidden field in other JSP.

```
<spring:hidden path="invalidData" cssClass="error"/>
<spring:errors path="invalidData" cssClass="error"/>
```

**LoginValidator:**

```
if(!(user.getUserName() != null &&
user.getUserName().equals("yunus") &&
user.getPassword() != null &&
user.getPassword().equals("yunus123"))){
errors.rejectValue("invalidData",
"username.or.password.invalid");}
```

**Another method:**

```
<!-- core tag lib with EL expression -->
<c:if test="${not empty error }">
<springtag:message code ="${error }"/>
</c:if>
```

**@Service**

```
public class UserService {
public boolean validate(User user){
if(!(user.getUserName() != null &&
user.getUserName().equals("yunus") &&
user.getPassword() != null &&
user.getPassword().equals("yunus123"))){
 return true;
} return false;
}}
```

**LoginController:**

```
else if(US.validate(user)) {
mv.addObject("error", "username.or.password.invalid");
redirectToLogin(user, mv);
}
• HTTP is stateless as it doesn't remember previous
request.(.jsp results)
```

**Alternate: Session Object:** Store each page result in session obj and merge at the end. (multiple request)

**Alternate: Cookies:** automatically browser will post details related to field.

Username and Password are Invalid		
User Name	<input type="text" value="yunus"/>	
Password	<input type="password" value="...."/>	
<input type="button" value="Login"/>		

So, can we use just **@Component** for all the components for auto scanning

**@Component** – Indicates a auto scan component.

**@Service** – Indicates a Service component in the business layer.

**@Repository** – Indicates DAO component in the persistence layer.

**@Controller** – Indicates a controller component in the presentation

Spring "filtering" to scan and register components' name which matched defined "regex", even the class is not annotated with @Component. <context:include-filter type="regex" expression="com.mkyong.customer.dao.\*DAO.\*" />

**Spring filtering exclude:** exclude specified components, to avoid Spring to detect and register it in Spring container.

```
<context:exclude-filter type="regex" expression="com.mkyong.customer.dao.*DAO.*" />
```

# DAY 29 SPRINGS WEB - DELETE

## URL patterns:

### Relative Path:

- Navigating a both jsp page in same folder.

**Absolute Path:** redirecting to other application.

(Redirect to yahoo or facebook)

### ../header.jsp:

- Move out from the current folder like cd..
- a/b/c --> ../../x.jsp --> x.jsp is available in 'a'

**Success.jsp:** Combining of all JSP in single page.

```
<%@ include %>
```

or

```
<jsp:include />
```

- Request to JSP --> converts to servlet --> lifecycle of servlet takes place --> Response

**Static include** --> Before converting into servlet, the content is combined together and generated as a servlet.

**Disadvantage** - JSP page size is 64KB so it will fail to generate a servlet

**You can use Apache tiles:** e.g.: CNN news channels.

- A page contain 10 resources...
- Moving the JSP files into config files as XML file.

**Interview:** What are the types are involved in order to render the success view?

- Used Apache tiles but am not able to recollect the config but I know the purpose. Combining all pages. Apache tiles will give config files to maintain your views.

**Pagination:** show 10 records per page and click next for another 10 records like this.....

**WebApplicationContext:** Merge another XML config files

```
<import resource="JDBCBDSSApplicatioContext.xml"/>
```

**Delete a record:**

```
<a href="
```

<!-- C URL is used for JSTL tag lib to form the complete URL -->

## Delete all records:

```
<form method="post" action="
```

```
<c:forEach items="${lis}" var="stuDTO"> <!--
for(StudentDTO stuDTO: stuList) -->
```

```
input type="checkbox" value="${stuDTO.sno}"
name="sno" /></td>
<td>${stuDTO.sno}</td>
<td>${stuDTO.sname}</td>
<td>${stuDTO.course}</td>
<td>${stuDTO.fee}</td>
```

```
@Service
```

```
public class StudentService {
 // depending on Student database objects.....
```

```
 @Autowired
 private StudentDAO SDAO;
```

```
 public List<StudentDTO> getAllStudents()
throws Exception
 {
 return SDAO.getAllStudents();
 }
```

```
 public boolean deleteStudent(int sno) throws Exception
 {
 return SDAO.deleteStudent(sno);
 }
}
```

## Student Portal

INSERT EDIT SHOW ALL LOG OUT

All selected records are deleted				
Delete	Sno	Sname	Course	Fee
<input type="checkbox"/>	1	FIRST	MS	10000
<input type="checkbox"/>	2	SECOND	MS	20000
<input type="checkbox"/>	3	SECOND	MS	20000
<input type="checkbox"/>	4	FIRST	MS	10000

Copyrights @2015

Contact Us About

## Session.load()

If you are certain object exists in cache, avoid DB hit  
If object not found in cache, it throws an exception

## Session.get()

If you are not sure if object exists in cache, hit the DB to fetch data  
If object not found in cache, it returns null, so check for null

# DAY 30 HIBERNATE

Hibernate an open source Java persistence framework project.

In hibernate we can write HQL instead of SQL which save developers to spend more time on writing the native SQL.

## ORM - object relational mapping

- It follows JPA (java persistence API)
- An ORM tool simplifies the data creation, data manipulation and data access.
- It is a programming technique that maps the object to the data stored in the database.

JPA (interfaces) - Hibernate, Top Link, JDO - **ORM tools**

- **Hibernate eclipse plugin-** Powerful tool to perform database interactions.

### *Interview: Why we use Hibernate?*

**Advantages:** Mapping between tables and objects using xml

- Object(Read/write/update/delete)
- DB independent
- Connection and Transaction management
- Simplifies complex join
- Automatic table creation

### **Disadvantages:**

- Black box testing - controlling with SQL is not possible.
- Performance issue - if we miss config files it leads issue.
  - Have to learn new API
  - Slower than JDBC
  - Gets complicated when there is many-many relationships

**Hibernate configs:**  
--> hibernate.cfg.xml

--> hibernate mapping file/hbm.xml avoid by annotation

--> POJO

### Session

- \* 1st level cache
- \* Data is shared across txns within the same session

### Steps:

Configuration \* single txn

Sessionfactory  
session

### Cache -

temporary memory area,  
which is used to store and fetch data.

### SessionFactory

- \* 2nd level cache
- \* Data is shared across sessions
- \* bunch of txns

### Main Class:

#### //Create configuration object

```
Configuration CON = new Configuration();
```

#### //Load configurations

```
CON.configure("hibernate.cfg.xml");
```

```
SessionFactory SF = CON.buildSessionFactory();
```

#### //Open a session

```
Session session = SF.openSession();
```

```
Transaction TRANS = session.beginTransaction();
```

#### //DB operations

```
/*User user = new User("Yunus","yunus123");
```

#### //insert

```
user = new User("Irshad","irshad123");*/
session.save(user);
```

#### //Delete

```
/*
User user = new User("Irshad");
session.delete(user);*/
```

#### //Read

```
User user = (User) session.load(User.class, "Irshad");
System.out.println(user);
```

**Update:** Update is like edit . use to change the value of record.

#### //Update

```
user.setPassword("Zakira");
TRANS.commit();}
```

### hibernate.cfg.xml:

```
<mapping resource="com/test/pojo/User.hbm.xml"/>
 <mapping class="com.test.pojo.User"/>
```

**//Annotations go into GETTER methods.**

### *Interview: Lazy vs Aggressive Loading?*

Hibernate is behave lazy loading object whereas Springs load aggressively.

**LAZY** - calls a request and then fetch the data very lazy.

**AGGRESSIVE** - fetch the data right away don't be late.

Hibernate internally maintains a cache mechanism. It identify the duplicate elements.

- If there are two objects fetching same data but hibernate displays only one data.

User user = new User("Irshad");

User user2 = new User("Irshad");

This is called as **Optimization techniques**.

# DAY 31 HQL - RELATIONSHIPS

**Cache** - fetching a data from the memory.

- Once the emp object is generated from memory If it called again then it gets data from cache.

*Interview: what are types of implementing cache in Hibernate?*

**Two types of cache mechanism**

- First level - Used by Session - save/delete/update/load
- Second level or Query cache - Session factory object

- Hibernate is going to generate predefined SQL and connections.

**Session.load** - can read only one record.

- If record is not available it throw an exception

**Session.get** - same as load but it shows "null".

**Reading multiple records:**

**SQL** : SELECT \* FROM tablename;

**Hibernate : HQL (Hibernate Query Language) :**

From com.test.User;

HQL - not case sensitive same as SQL.

**SQL** : SELECT sno,sname FROM tablename;

**HQL** : Select stu.sno, stu.sname from com.test.Student  
as stu;

**//Hibernate Query**

```
Query query = session.createQuery("From
```

```
com.test.pojo.User");
```

```
List<User> userList = query.list();
```

```
for(User user : userList)
```

```
 System.out.println(user);
```

- When running Hibernate will execute as SQL statement in backend.

**For multiple:** // Contains object array - Hibernate will treat as objects.

```
List<Object []> userList = query.list();
```

```
for(Object obj : userList)
```

```
 System.out.println(obj);
```

- Hibernate will demand for a primary key in the table and in code ("@ID");
- If a table doesn't have a primary key, then hibernate combines the columns as key object.

**Combination of two columns**

- Composite primary key (SSN+driver's license)
- We have a new POJO class with @embedded of all columns.

**Relationships in Hibernate:**

Student --> sno, sname, courseID

Course --> courseID, Cname

- Course is a primary table whereas Student is child table with a foreign key courseID.

Create tables course, student, address.

- In student click foreign keys tab and FK\_cid, then reference to table and column id.

**One to many relationship:**

```
<one-to-many class="com.test.demo.pojo.Student" />
```

- One course can register by multiple students....

**In course.java:**

```
private Set<Student> students = new
HashSet<Student>(0);
```

**In student.java:**

```
private Set<Address> addresses = new
HashSet<Address>(0);
```

**Many to one relationship:**

```
<many-to-one name="course"
```

```
class="com.test.demo.pojo.Course" fetch="select">
```

- Multiple students can register one course.

```
<many-to-one name="student"
```

```
class="com.test.demo.pojo.Student" fetch="select">
```

- Multiple address can be pointed to one student.

```
<set name="addresses" table="address" inverse="true"
lazy="true" fetch="select">
```

- One table has 200 tables, fetch one table data so highly recommend to go with Lazy.

- If we need to fetch all 200 tables' details, I can go with aggressive.

**Joins:** fetch all the three table details instead of writing 3 SQL statements. fetch type = "join";

*Interview: Do you know about Select (N+1) issue?*

- If Relationships are not properly configured, hibernate make select statements.

**Cascade:**

- If you make changes in parent table, changes must applied in child table. cascade = "delete, update, all" delete\_rule = 'CASCADE'

**business object (BO)** is used to store the project's business function, the real database operations (CRUD) works should not be involved in this class, instead it has a DAO class to do it.

**SPRING HIBERNATE TABLE CONFIG** <class name="com.mkyong.stock.model.Stock" table="stock" catalog (db name)="mkyong">

<id name="stockId" type="java.lang.Long">

# DAY 32 HIBERNATE - SPRINGS

## HibernateRelationApp:

```
Student stu = (Student) session.load(Student.class, 1001);
System.out.println(stu.getSname());
Set<Address> address = stu.getAddresses();
for(Address addr : address)
 System.out.println(addr.getState());
```

- Fetch address city from the student number.
- Change in java programs of student

```
@OneToMany(fetch = FetchType.EAGER, mappedBy =
"student")
<set name="students" table="student" inverse="true"
lazy="false" fetch="join">
```

## Interview: what is criteria?

### Are you comfortable with criteria or HQL statement?

Criteria is an object which completely avoid HQL statements. Take care of pagination records.

```
Criteria crit = session.createCriteria(Student.class);
crit.add(Restrictions.between("sno", 1002, 1004))
// where clause between
.addOrder(org.hibernate.criterion.Order.desc("sno"))
// descending the records
crit.setMaxResults(2); // 2 results per page
crit.setFirstResult(2); // next 2 results after Old 2
```

## Hibernate Inheritance:

- In 2 tables have same sno, sname. It is repeating, so create one pojo class and execute as Table1 extends Table.

## Interview: Have you used Spring with JPA?

Internally we are using Hibernate.

## JPA: Java Persistence API

- (JPA) is a Java application programming interface specification that describes the management of relational data in applications
- Switching to new ORM, if you used JPA then just use change config files.
- JPA is easy in migrating process.

## Advantages of Spring with Hibernate:

Use Annotations: @Component @service @controller in springs

@repository in Springs hibernate

- Helps in deleting the connection code in Hibernate APP

- Spring ORM lib is needed.

- Inject Datasource

```
<bean id="sessionFactory" class=
"org.springframework.orm.hibernate4
.LocalSessionFactoryBean">
<property name="mappingResources">
<value>com/test/demo/pojo/Student.hbm.xml</value>
```

- Copy your Student pojo class and resources (xml) into Spring project

```
<property name="hibernateProperties">
hibernate.dialect=
org.hibernate.dialect.MySQLDialect
```

## SpringDemo:

```
ApplicationContext AC =
new ClassPathXmlApplicationContext
("HibApplicationContext.xml");
```

```
SessionFactory SF =
AC.getBean("sessionFactory",Session
Factory.class);
```

```
Student STU = new Student(666,"Rajini");
```

## Hibernate-Springs-DAO:

### StudentDAO:

```
@Component
public class StudentDAO {
 @Autowired
 private SessionFactory SF;
```

```
public List<Student> getallStudents()
 Session session = SF.openSession();
 Criteria crit = session.createCriteria(Student.class);
 return crit.list();
```

```
StudentDAO stuDAO = (StudentDAO)
AC.getBean(StudentDAO.class);
```

```
List<Student> stulist = stuDAO.getallStudents();
for(Student stu : stulist)
 System.out.println(stu);
```

**Model, Business Object (BO) and Data Access Object (DAO) pattern** is useful to identify the layer clearly to avoid mess up the project structure.

## Spring-Hibernate Integration : 2 ways to connect Spring and Hibernate

- IOC with Hibernate template (there is a Spring JDBC template also, for doing DB txns via JDBC)

- Extend HibernateDAO Support and AOP o Configure HibernateSessionFactory o Extend the DAO implementation of the Hibernate transaction support using AOP

# DAY 33 JSON – XML - XSD

**XML/ JSON Format** -- returning a data

**Transferring of data between APPs:**

- XML - takes 10KB
- JSON - takes 6KB

**XML:**

```
<Response>
<RegionName>Florida</RegionName>
<City>Melbourne</City>
</Response>
```

**JSON:**

```
{"region_name": "Florida", "city": "Melbourne"}
```

- Generating a view UI with data generated from XML/JSON services

All applications are called services because in future you can switch into various languages like from JAVA to PHP and fetch the data.

**XSD:**

XML -- validation of data is done by XSD.

- Whether XML data is XML and JSON data is written in JSON Consumer

Service controller - written in WADL//WASDL Provider

**XSD Namespaces:**

- If there are two city field you can relate using namespaces.

<City>Melbourne</City> is written as <Country:City>

<City>Jupiter</City> is written as <Country2:City>

**XSD/XML Mapping:**

Student.xsd:

```
targetnamespace:"URL"
<student element>
<address element>....
```

**Another XML/XSD:**

```
xmlns:stu="URL"
schema: "URL"
```

If there is no namespace such as xmlns = it is a default namespace

**Packaging** = into jar files [XSD files + JAXB files]

**Maven advantages:**

1. Run the plugin information
2. You can use this maven project in order project as a dependency

**Interview: What are the parsers you used to read/write the XML data?**

- **DOM - Document object model** - load a XML file in memory and do manipulations.

\* ADD/UPDATE/DELETE

\* Disadvantage: Read a 10 GB file it will fail

- **SAX Simple API for XML)parser** –

It is an event driven "some content is changed and updated", read the data.

\* XPATH (XML location PATH) - Employee

/Address/City information

**JAXB - Java API for XML binding** contains setters/getters it is also called Hibernate POJO

- Contains marshalls and unmarshalls.

JAXB Class is used, this can be automatically generated by maven plugin.

**Student.xsd:**

```
targetNamespace="http://www.test.com/"
 xmlns:tns="http://www.test.com/"
elementFormDefault="qualified">
<element name="Student">
<complexType> // You have multiple elements
<sequence> // order
<element name="address" type="tns:Address">
<complexType name="Address">
<sequence>
<element name="city" type="string"/>
```

- Build the XSD files and generate JAVA classes from Maven plugin.

**Interview: How have you generate JAXB classes?**

- We have written XSD files and generated JAXB plugin and generate java classes.

**Run - Maven generates sources**

JAXB files gets generated in target/generated-sources.

**You can use BusinessObjects as dependency:**

first 4 lines in pom.xml

```
<groupId>com.business.objects</groupId>
<artifactId>com.rest.services</artifactId>
<version>0.0.1-SNAPSHOT</version>
```

**Marshalling:**

- Conversion of java object into XML or JSON
- ```
//Student student = new Student(456,"Jhon",
4500,new Address("Plymouth", "MN", "55447"));
JAXBContext jc = JAXBContext.newInstance("com.test"); // JAXB is a heavy weight
/*Marshaller marshaller = jc.createMarshaller();
marshaller.marshal(student, System.out);*/
```

UnMarshall:

- Conversion of XML/JSON into java object.
- ```
Unmarshaller unmarshaller = jc.createUnmarshaller();
Student student = (Student)unmarshaller.unmarshal(new File("G:\\JAVA-J2EE Workspace\\XMLAndJson\\src\\main\\resources\\student.xml"));
```

**Spring OXM:**

- Object XML mapping maintains the marshalling/unmarshall.

**Converting into JSON:**

```
Gson gson = new Gson();
String json = gson.toJson(student);
System.out.println("Json " + json);
```

**OUTPUT:** Json { "sno":456,"sname":"Jhon","course":4500.0,"address":{"city":"Plymouth","state":"MN","zipcode":"57"}}

A web service is a kind of software that is accessible on the Internet. It makes use of the XML messaging system and offers an easy to understand, interface for the end users. Web services transfer/receive messages to/from application respectively, via HTTP protocol. It uses XML to encode data.

# DAY 34 REST

## Disadvantages of web services:

- Need to know All data types to be XML types

### Web services

- Reusable component data to file (XML,JSON,tst,csv...) overhead

Distributed transaction mgmt is not possible. E.g. there are 3 method calls and method2 call is to a webservice. If method 3 fails, then there is no way to tell the webmethod to rollback.

### Two Types

- REST \* SOAP

### REST Frameworks:

- Spring MVC (JAX-RS API) by default supports REST based services
- (Apache CXF RestEasy)

### Advantages of REST:

- Light weight service by using HTTP methods you get request/response(GET,POST,PUT,DELETE)

### WebAppContext.xml:

```
<!-- Rest Services -->
 <bean
 class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter">
 <property name="messageConverters">
 <list>
 <!-- Marshal and unmarshal -->
 <ref bean="xmlConverter" />
 <ref bean="jsonConverter" />
 </list>
 </property>
 </bean>
```

```
<!-- XML Mapping - sending a XML data it is going to use XML converter -->
 <bean id="xmlConverter"
 class="org.springframework.http.converter.xml.MarshallingHttpMessageConverter">
 <constructor-arg ref="jaxbMarshaller" />
 <property name="supportedMediaTypes"
 value="application/xml" />
 </bean>
```

```
<bean id="jaxbMarshaller"
 class="org.springframework.oxm.jaxb.Jaxb2Marshaller">
 <property name="classesToBeBound">
 <!-- internally it is going to bound JAXB classes -->
 <value>com.test.Student</value>
 </property>
 </bean>
```

<!-- JSON Mapping used when client is posting a JSON data and getting a response-->

Write XSD --> generate JAXB classes -->  
Request/Response

### Advantages of web services:

- JAXWS has API to access the header info
- Stateless so only one instance available for all callers, so thread safe

### RestService.java:

```
@Controller
@RequestMapping("/stuservice")
public class RestService {
 @RequestMapping(method=RequestMethod.GET,
 value="/{sno}")
 public @ResponseBody Student
 getStudent(@PathVariable Integer sno){
 // Returning a java object back..
 Student student = new Student(456,"Jhon",4500,new
 Address("Plymouth", "MN", "55447"));
 return student;
 }
}
```

**Springs RestFul services:** Inside of Model and view we use java object as Response Body.

### Download Advanced REST client:

#### Send the URL:

<http://localhost:8080/SpringMVCWebAppREST/mvc/stuservice/101> then type: Accept: application/JSON or XML to be converted....

- Client is browser and asks as XML and then it returns back the data in XML format.

**POST:** Posting an XML/JSON data to server from client.

#### Click POST in APP:

```
Accept: application/XML
 // what type client wants from the server.
Content-Type: application/XML
 // what type client is sending to server
```

### REST service - Spring MVC

- Client is going to request and response in terms of objects.

### RestTemplateApp:

```
ApplicationContext springContainer = new
ClassPathXmlApplicationContext("restContext.xml");
RestTemplate restTemplate =
springContainer.getBean("restTemplate",RestTemplate.class);
Student student = (Student)
restTemplate.getForObject("http://localhost:8080/Spring
MVCWebAppREST/mvc/stuservice/101",
com.test.Student.class);
Response response = // Reading object from website
restTemplate.getForObject("http://freegeoip.net/xml/",
Response.class);
```

### JAX WS

Supports asynchronous and message oriented web services

100% XML with JAXB

Uses annotations and Deployment Descriptor is optional

### JAX RPC

Cannot support asynchronous and message oriented web services

Uses java type binding and only 90% XML

No annotations and Deployment Descriptor is must

# DAY 35 XSD - APACHE CXF

*Interview: How do you implemented a security for your services?*

- OAuth - open authentication - userID/tokenID store in your device --> OAuth service (GOOGLE) --> Gives Access Token --> make an access to service
- Spring security
- Basic authentication - username/password verification browser dialog box

Spring MVC doesn't follow JAX-RS standards.

Few Web services frameworks are Apache CXF, REST Easy

Apache CXF is an open source services framework. CXF helps you build and develop services using frontend programming APIs, like JAX-WS and JAX-RS.

- Framework will read XML/JSON data and convert (marshall/unmarshall) into java object.
- XSD is used to define rules how XML is written. XSD is shared to non-java client.
- By using XSD we generate java objects using Maven JAXB plugin. → Run as Maven generate resources.
- JAXB - advantage is we are sharing a java object in terms of XML. (marshall/unmarshall)
- In Real time, we write XSD files and then we use the objects throughout our APP.

In project --> src resources --> create XSD folder  
--> Student.xsd

```
<element name="Student"> // we cannot reuse it.
<complexType> // reuse this elements
<sequence>
 <element name="sno" type="int"></element>
<element name="address"
type="tns:Address"></element>
```

You can use Student.xsd components by declaring "student.xsd" in namespace.

Maven Install → generate the project into LIB. [BusinessObjects.jar] and first 4 lines of pom.xml as maven dependency.

Business objects → XSD to JAXB files.

Service Layer -- CXF Frame -- JAX-RS

- First write an interface with service methods. @PATH/@HTTP Req/Response
- Configure your implementation as a spring bean.

web.xml:

```
<servlet>
 <servlet-name>CXFServlet</servlet-name>
 <servlet-class>org.apache.cxf.
transport.servlet.CXFServlet</servlet-class>
```

```
@Produces({MediaType.APPLICATION_XML,
MediaType.APPLICATION_JSON})
public class StudentService {
```

```
 @GET
 @Path("/student/{sno}")
 public Student getStudent
(@PathParam("sno") int sno){
 Address address =
new Address("Plymouth", "MN", "45678");
 Student student =
new Student(101, "Jhon", 4500, address);
 return student;
```

APPContext.xml:

```
<!-- CXF Resources -->
<import resource="classpath:
META-INF/cxf/cxf.xml" />

<bean id="studentService" class=
"com.test.service.StudentService" />
<jaxrs:server id="appServices" address="/">
 <jaxrs:serviceBeans>
 <ref bean="studentService" />
 <ref bean="creditService" />
```

# DAY 36 APACHE CXF

**web.xml** - CXF Servlet contains spring functionalities.

**Pom.xml:**

<properties>

```
<cxn.version>2.5.1</cxn.version>
<commons-logging-version>1.1.1</commons-logging-version>
<org.springframework.version>4.0.0.RELEASE</org.springframework.version>
</properties>
```

**Jetty Server:** download eclipse plugin

Project --> Run jetty --> New --> Run

**CXF framework** automatically generate a WADL --> helps in understanding for non-java client.

**Requesting a WADL:**

http://localhost:8080/cxfservices/api?\_wadl

Take requirement → **Service contract (WADL)** share to client and then implement.

CXF --> WADL --> REST

--> WSDL --> SOAP

<element name="providers" type="tns:Provider" **minOccurs="0" maxOccurs="unbounded"/>**

// min occurs means **minimum and unbounded is maximum providers. Infinity**

@Path("/banking") -- sub level in the APP.

**Generate XML from XSD**

- Right Click → generate → XML → copy and paste the XML content in Advanced REST client.

The screenshot shows the Advanced REST Client interface. The URL is set to `http://localhost:8080/cxfservices/api/banking/creditreport`. A POST method is selected. The Headers tab shows `Accept: application/json`, `userid: jhon`, and `password: test1234`. The Payload tab contains the following XML payload:

```
<?xml version='1.0' encoding='UTF-8'?>
<tns:CreditRequest xmlns:tns="http://www.test.bank.com/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.test.bank.com/ creditreport.xsd">
 <tns:ssn>7655-545-56454</tns:ssn>
 <tns:fname>yunus</tns:fname>
 <tns:dob>002-04-1991</tns:dob>
 <tns:zipcode>32901</tns:zipcode>
</tns:CreditRequest>
```

The Response tab shows the JSON output:

```
{
 "noOfAccounts": 5,
 "creditScore": 765,
 "providers": null,
 "zipcode": "45678",
 "address": {
 "city": "Xenium",
 "state": "NJ",
 "zipcode": "45455"
 }
}
```

# DAY 37 APACHE CXF

**CXF Servlet** - will call CreditService and gives response to StudentService.

```
public class ExceptionHandler implements ExceptionMapper<Exception>
 // provided for CXF framework.
public Response toResponse(Exception exception) {
 ResponseBuilder rb = Response.status(Response.Status.INTERNAL_SERVER_ERROR);
 Status status = new Status("Error", "Something went wrong while processing your request.. Try Again/Conatct Admin");
 rb.entity(status);
 return rb.build();
```

**Interceptors in Springs:**  
preHandle() – fired before request is processed  
postHandle() – fired after request is processed  
afterCompletion() – fired after view is rendered

## Data Validations:

**Request Interceptor** - intercept your request and respond to it before invoking the service

**Response Interceptor** - any extra manipulations to be done before responding to client.

```
public class CustomRequestInterceptor implements RequestHandler{
 public Response handleRequest(Message message, ClassResourceInfo resource) {
//Read Header Data
Map<String, List<String>> headers = CastUtils.cast((Map)message.get(Message.PROTOCOL_HEADERS));
System.out.println("Headers ##### " + headers);
String queryString = (String)message.get(Message.QUERY_STRING);
System.out.println("queryString" + queryString);
return validateRequest(headers); //rb.build(); //FWD request to actual service.
```

```
public Response validateRequest(Map<String, List<String>> headers){
 try{
 //get userid
 String userID = headers.get("userid").get(0);
 String password = headers.get("password").get(0);
 if(userID != null && userID.equals("jhon") && password != null && password.equals("test1234")){
 return null; //login success
 }else{
 return buildErrorMessage("Login Failed!!!", "Please Validate UserName & Password");
 }
 }catch(Exception ex){
 return buildErrorMessage("Login Failed!!!", "UserName & Password are misising in the Http Header");
 }
}
```

```
public Response buildErrorMessage(String stausCode, String msg){
 ResponseBuilder rb = Response.status(Response.Status.INTERNAL_SERVER_ERROR);
 Status status = new Status(stausCode, msg);
 rb.entity(status);
 return rb.build();
```

**Basic authentication** is userid and password in Header column "Advanced Rest Client"

**OAuth implementation** is 99% companies.

**SOAP:** Simple Object Access Protocol allows communication between applications located on remote machines. It in turn uses transport protocols like HTTP, SMTP, and FTP.

# DAY 38      SOAP

SOAP is an XML based protocol to transfer between computers.

## Interview: Diff between REST vs SOAP?

- REST is a lightweight with HTTP methods. REST contains different request format such as JSON, XML and Objects. **Uses HTTP (GET/POST/DELETE)**
- **SOAP structure** - heavyweight services such as Request/Response XML. **Uses HTTP POST only**
- Follows a web security standards.

## WSDL - XML/XSD

### WADL - Web APP description language

- Checks what request/response is stored here?
- If service need authentication then use this,  
`<soap:Envelope>`  
    `<soap:header>                <Auth>`
- If service occurs an error then use this,  
`<soap:body>            <SOAP: Request> and <SOAP: Response>`  
    `<SOAP:fault>`

### XML digital signatures:

- If there is a document which is very important, then sign the document by running an algorithm using private key and public key.
- **Ex:** Transfers funds → SRC - sign - generate a unique key (hashcode) for that data → DEST

## Interview: How you provided security for your web services?

- We used normal username and password process passed under SOAP header.

### WSDL: SOAP based services.

- Java to WSDL - write java program and then implement WSDL file
- 99% we are going to follow WSDL file and then java implementation.
- WSDL is a service contract. - Write the WSDL then share to the clients, if client agrees then start java implementation.

### REST based services

 then generate WADL.

- **SOAP UI**
- **SOAP over HTTP** (prepare your SOAP request and post to the end point HTTP)

**Java client** - Request and access a service.

**TCP/IP monitor** is used to debug the SOAP web services.

**Enumeration:** is an interface which allows sequential access to all the elements stored in the collection.

## Ex: free-web-services.com-->Unit -->currency converter. → You can view the WSDL file.

Data types, pojos, XSD elements. All XSD types can be declared in separate file and then linked into WSDL

**<WSDL types>** - schema definition for input and output  
**<WSDL enumeration>** -- enter only one value (unique values)

- ENUM {"JAN", "FEB", "MAR"} → comparing with String month, if it is equal.

## StudentService //JAVA CODE OBJECTS

**studentResponse updateStu(StudentRequest)**  
Message types based on XSD types for REQUEST, RESPONSE and FAULT

**WSDL message:** going to bind the parameters

Links operations and message types. -request/response/fault  
**WSDL portTypes:** differentiate the operations such as UpdateStu with Student Req/Res.

- **Protocol and data format for particular port type**
- **WSDL bindings:** use HTTP protocols, sometimes we use synchronize service using JMS. Sending a message but don't wait for response.

- Once the binding is done then moves to **SERVICE END point as URL**.

**Service:** Binds java service to port, this is where you define soap address(end point)

**Convert into WSDL:** import this XSD file in WSDL

`<StudentResponse>`  
status code  
status message

- We have tools to generate WSDL files.  
\* **SOAP UI plugin** → Install SOAP in eclipse.

## SOAP OVER HTTP:

- Take the URL in END point on WSDL and send to java project as a HTTP connection.
- Send SOAP request

## JAVA client:

 How to write it?

- Take a WSDL and generate proxy classes using WSDL to java classes use MAVEN CXF java plugin.

## How to develop a service itself?

1. WSDL to java by generating proxy classes
2. instead of client program we use @Web service interface
3. Impl(Service Provider)
4. Java to WSDL → java implementation → CXF-WSDL.

### SOAP Faults:

**Version Mismatch:** Invalid namespace in envelope

**Client:** Client incorrectly formed the SOAP message

**Must understand:** SOAP processor is unable to process header

**Server:** Server cannot process SOAP message

**Bottom Up(Java first) Top Down(WSDL first)** – write Java interface, use WSDL annotations(JAXWS) and then generate WSDL  
\* Requires less knowledge of WSDL, XML, XSD  
\* Interoperability issues may occur with consumers/providers written in other languages  
\* If end point interface changes, WSDL changes , and client to be rewritten  
\* Requires more knowledge of WSDL, XML, XSD  
\* Better interoperability  
\* Less sensitive to change

# DAY 39 WSDL ← → SOAP

**WSDL:** To test your service use SOAP UI...

- Import the WSDL into SOAP UI and give inputs and execute.
- In resources, copy the WSDL file and then run as Maven generate sources.
- Using WSDL generate proxy classes.
- Once the code goes into the Production service "CLIENT" from Test service.

**SOAP Client Program:**

```
public class CurrencyClient {

 public static void main(String[] args) {
 //Create Web service Client Object
 CurrencyConvertor currencyConvertor =
new CurrencyConvertor();
 CurrencyConvertorSoap service =
currencyConvertor.getCurrencyConvertorSoap();
// In WSDL check for bindings....
 double conversionRate =
service.conversionRate(Currency.USD, Currency.INR);
 System.out.println(conversionRate);
 }
}
```

**CXF client** creates a SOAP request and posts the request to service "Remote Server" and service returns SOAP response.

**Debugging:**

- TCP monitor acts as a call forwarding between Client and server.

Add → Http port as 9999 → webservices.net → start New service created at port 9999:

```
<wsdl:service name="CurrencyConvertor">
 <wsdl:port name="CurrencyConvertorSoap"
binding="tns:CurrencyConvertorSoap">
 <soap:address
location="http://localhost:9999/CurrencyConvertor.asmx
 "/>
```

@WebService name - <wsdl:portType name=portName - <wsdl:port name=serviceName - <wsdl:service name=targetNamespace – namespace for <wsdl:portType endpointInterface – complete name of service endpoint wsdlLocation – location of pre-defined WSDL  
If class has @WebService then all public methods are automatically web methods, no need to explicitly put @WebMethod for each method.

**In TCP monitor:** it displays

```
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ConversionRate
xmlns="http://www.webserviceX.NET/">
<FromCurrency>USD
<ToCurrency>INR
```

- Java Application posted a SOAP request and get the SOAP response

```
<ConversionRateResult>63.57
Class will have @WebService and method will have @WebMethod
```

**Converting a Apache CXF service into SOAP web service:**

@WebService      **JAXWS:** Java API for XML based web services.  
public interface ICreditSOAPService {  
 CreditResponse getCreditReport(CreditRequest  
request);      JAXWS specification is used, it is pre-included in J2EE. Once you have application server plugin, the required jars are auto included by application server.

**In applicationContext.xml:**

Add bean id and

```
<jaxws:endpoint id="creditSoapService"
implementor="#creditSoapServiceBean"
address="/creditService" />
```

**Interview: Can you tell me what the implementations of a service?**

- \* write a interface
- \* interface is annotated with @webService
- \* write implementation class with request objects
- \* configure implementation as a spring bean.
- \* provide a end point URL
- \* Use this end point URL to generate WSDL file and share to client.

**My WSDL:**

http://localhost:8080/cxfservices/api/creditService?wsdl

**// with end point...**

**Test this service:**

- \* SOAP UI

**JavaScript** is a client-side scripting language that can be inserted into HTML pages and is understood by web browsers.

An **alert box** displays only one button which is the OK button.

But a **Confirmation box** displays two buttons namely OK and cancel.

# DAY 40 AJAX - JQUERY

## ASync JavaScript with XML:

With AJAX, the JavaScript does not have to wait for the server response, but can instead:

- execute other scripts while waiting for server response
- deal with the response when the response ready

## Interview: How you implemented AJAX and pass the data?

- AJAX is used to autopopulate data in text field

Eg: If I enter EmpID, it must display name and salary in text field automatically.

- AJAX is the art of exchanging data with a server, and updating parts of a web page - without reloading the whole page.
- It is a performance issue when we refresh whole page
- AJAX is implemented on Client side/browser which is written in JavaScript.

## AJAX.jsp:

```
<script type="text/javascript">
function getEmployee()
 alert("Testing..."); //.... AJAX script
<form> NAME: <input type="text" name="name"
onchange="getEmployee()"/>
// id is used to read the element.... onchange is a javaSCript
event
```

## XMLHttpRequest object:

- It is used to exchange data with a server behind the scenes.
- Check if the browser supports the XMLHttpRequest

```
var xmlhttp; if (window.XMLHttpRequest)
{ // code for IE7+, Firefox, Chrome, Opera, Safari
xmlhttp=new XMLHttpRequest();
else // code for IE6, IE5
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
```

**Send a Request To a Server:** There are two options:

- xmlhttp.open("GET", "ajax\_info.txt", true);
 //url: the location of the file on the server
 //async: true (asynchronous) or false (synchronous)
- xmlhttp.send(); //Sends the request off to the server.
- xmlhttp.send("fname=Henry&lname=Ford");
 //string: Only used for POST requests
- POST is more robust and secure than GET

**setRequestHeader(header,value)**

//Adds HTTP headers to the request. // header: specifies the header name // value: specifies the header value

```
xmlhttp.setRequestHeader("Content-type","application/xml");
```

**Async=true**

- Specify a function to execute when the response is ready in the onreadystatechange event

```
xmlhttp.onreadystatechange=function()
if (xmlhttp.readyState==4 && xmlhttp.status==200)
{
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
```

## Server Response:

Two Property of XMLHttpRequest object

- **responseText** - get the response data as a string
- **responseXML** - get the response data as XML data

## Callback function:

- A function passed as a parameter to another function. If you have more than one AJAX task on your website, you should create ONE standard function for creating the XMLHttpRequest object, and call this for each AJAX task.

```
function DisplayStudent()
```

```
sno = document.getElementById("snoID").value;
xmlhttpReq = getXMLHttpRequest();
xmlhttpReq.onreadystatechange()
```

**// RESPONSE from XML content**

```
ajaxResponse = xmlhttpReq.responseXML;
alert/ajaxResponse);
```

**// Displaying content in other fields using number.**

```
document.getElementById("snameID").value =
ajaxResponse.getElementsByTagName('sname')[0].childNodes[0].nodeValue;
```

URL =

```
"http://localhost:8080/SpringMVCWebAppREST/mvc/stuuser
vice/"+sno+"/"
xmlhttpReq.open("GET",URL,true);
xmlhttpReq.setRequestHeader("Content-
type", "application/xml");
```

```
xmlhttpReq.send(); jQuery is a lightweight, "write less, do
more", JavaScript library. jQuery also
simplifies a lot of the complicated things
```

## JQUERY:

- It is a JavaScript lib. DOM manipulation.
- Download and transfer to project folder.

```
<script type="text/javascript" src="myScript.js" ></script>
<script type="text/javascript" src="jquery-1.11.3.js"
```

**myScript.js:**

```
function DisplayStudent()
```

```
{ $.ajax({
 type: "GET",
 URL:
"http://localhost:8080/SpringMVCWebAppREST/mvc/stuuser
vice/"+$("#snoID").val()"/",
 dataType:"xml"
 }).done(function.ajaxResponse)
 { alert/ajaxResponse);
 $("#snameID").val($(ajaxResponse).find("sname").text());
}); }
```

# DAY 41 Project Architecture

*Interview: Diff between Web server and Application Server*

Web Server -	JSP/SERVLET API	TOMCAT/JETTY
Application server -	Web server + [JMS, JNDI, FTP/SMTP, EJB]	JBOSS, WebLogic, WebSphere

- Application is generic, so you can use any server.

*Interview: Can you explain me the architecture of your project?*

## 1. (BackEnd) DB Layer - SQL - SELECT/JOIN

- Used Hibernate which doesn't allow us to write SQL commands.

## 2. DAO Layer - Read/write/update operations related to the DB layer.

- DAO is an object that provides an abstract interface to database mechanism. BasicDataSource interface.
- We use connection pooling mechanisms using Apache DBCP Lib or C3Po Lib.
- We also use Jboss/Weblogic server for the connection of layers.
- 70% we use JNDI for the connection.
- AOP concept is used to handle the DAO exceptions such as database or connection.
- We are not supposed to write any business logic in this layer.

## 2. Data Service layer - DOTNET is planning to reuse the logic.

- So I can implement DAO layer as a service. This is called SOA.
- According to my project, I have implemented REST services to expose my DB operations.
  - E.g.: Wells Fargo gets credit score from Discover services.
- Client receives a result which has a Combination of DAO and other external services
- This layer services can interact with backend systems such as SOAP and REST or may be JMS.

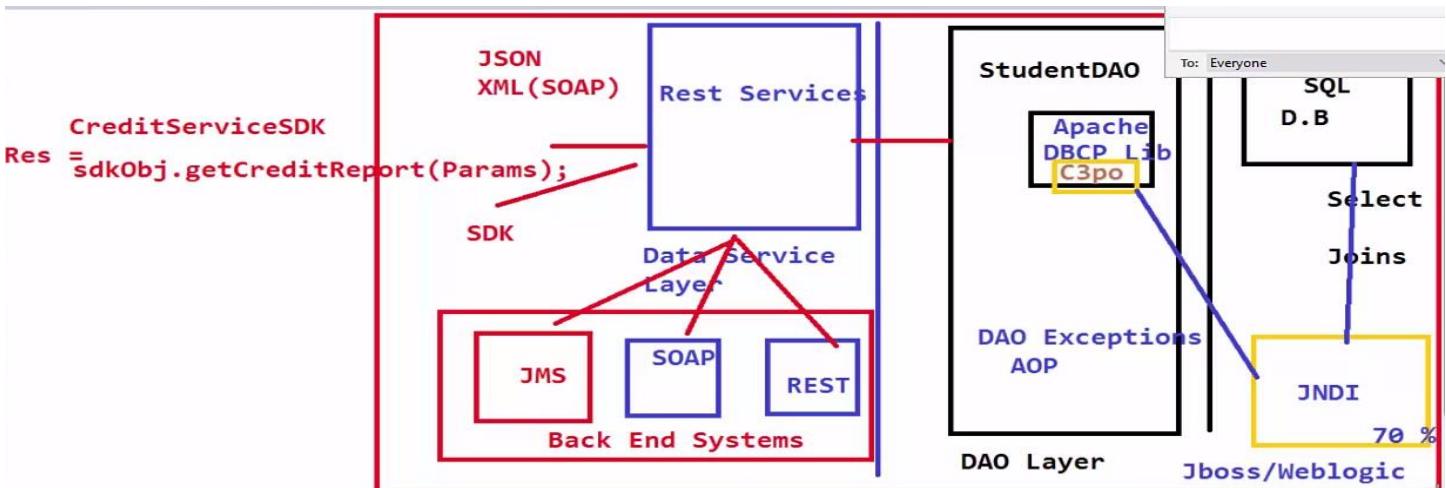
## 4. SDK - Service development Kit

- It is a Class, Client will create an object Handover the service to the client.

```
SDKobj.getCreditScore(params);
```

"Business delegate" design pattern. - SDK makes a REST call using APACHE CXF framework.

- Service WebAPP: This LIB is only used by java client.



## 5. UI Layer: Presentation layer interact with the REST Service layer

- **Static content** - HTML, JavaScript, Jpg, CSS
- **Browser/Client** - static content is redirecting to web server is done by static Web server "Apache HTTP server" which maintains load balancing mechanism.
- **NGINX server** - advance then Apache HTTP server but both serve same static.
- **99% of the APPS will hit static server** and return back the content.
- **APP is down and under maintenance** - you can display a page via static server.
- **Content server** - If client changes logo and content, then you might use content server "collection of HTML or jpg files".
- **Static Web Server** - can perform basic authentication such as username and password.
- **Site Minder** - provides security to your APP. 99% we use, because they might use password challenge.
- **SSO applications** - Single sign-on (SSO) is a property of user logs in once and gains access to all systems without being prompted to log in again at each of them.
  - **Example:** Google login but access to all google products GMAIL, YOUTUBE, GOOGLE+ etc.
  - If a person access to GMAIL directly then site minder is used to protect this sub APP.

All sub applications is protected by Site Minder.

A protected method may only be accessed by classes or interfaces of the same package or by subclasses of the class in which it is declared.

- User is logged in and authenticate then site minder generates a SESSIONID
- **LDAP server configuration**

- is authentication server username, password and roles such as this user can access this sub apps and he can use 60days.

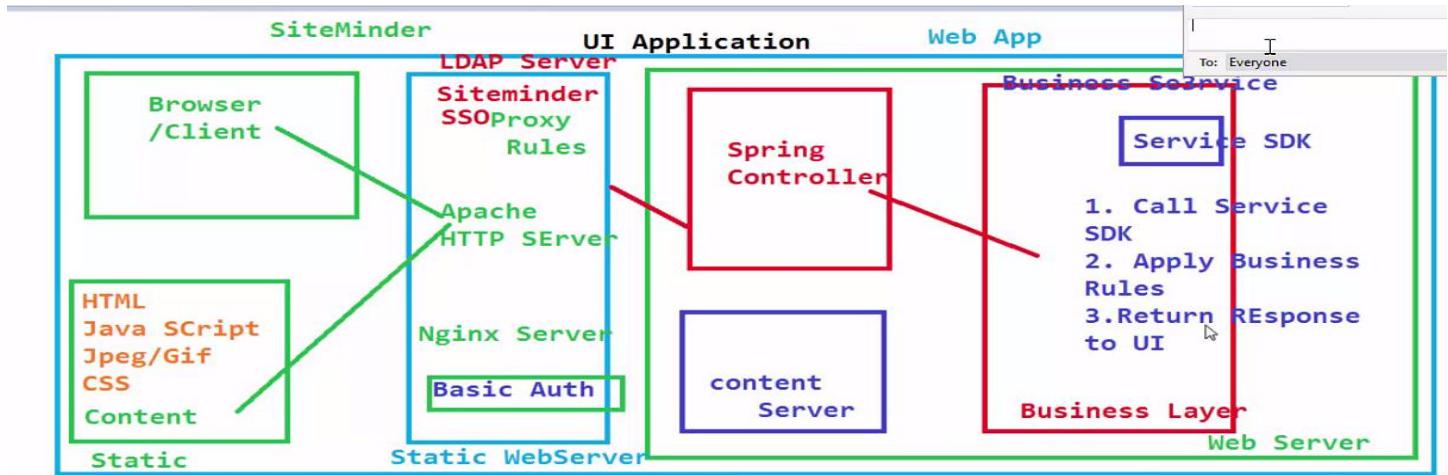
- improves the performance. We have used LDAP server to store the username and password.

\* Using Spring, you can interact java code with the LDAP server.

- **Spring Controller** - we have used Spring MVC layer for implementation of UI layer.

## 6. Business layer - spring controller captures the authenticate response and then move to business layer.

Call the backend services SDK. → Apply business rules. → Return response to UI.



**Interview: What are the necessary steps you will take care when you design an APP?**

- For future enhancements, if we require REST instead of SOAP services, then you can switch with minimal impact. If we are sharing the same objects in two layers and then there will be impact.
- Application must be more flexible.

five of those SDLC models, namely; Waterfall model, Iterative model, V-shaped model, Spiral model, agile model.

# DAY 42 Version Control - SDLC

**Interview: How do you or where do you maintain the content server?**

- Oracle content server, Adobe, IBM      Open Text - "Vignette Content management server" –
- 99% use this - contains HTML, jpg etc.      Site channel instead of URL.      Terms/Conditions content.

## Project Implementation:

- Split into multiple java projects.

## BusinessObjects:

- Write XSD and convert into JAXB files.      \* Share XSD with Client via WSDL/WADL.
- Maven project with group id and artifact id.

```
<element name="IPDetails">
 <complexType>
 <Sequence>
 <element name="ip" type="string"/>
```

## StudentServices:

- Write only interfaces use the business objects request and response.
- If java client, then we can share BusinessObjects and StudentService.

```
@Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
```

```
public interface IIPDetailService {
 @GET
 @Path("/ipdetails")
 IPDetails getIPDetails();
```

## StudentCore:

- Implementation of interfaces      \* DAO package

```
@Component
```

```
public class ExternalIPDetailsSDK {
 @Autowired
 private RestTemplate restTemplate;
 public IPDetails getIPDetails(){
 IPDetails ipDetails = restTemplate.getForObject("http://freegeoip.net/json/",IPDetails.class);
 return ipDetails;
 }
}
```

## StudentSDK:

- Planning to expose everything as a service.
- Client classes in terms of SDK instead of WSDL or WADL using CXF, Restful frameworks.

## WebApplication:

**Version control** - maintain your changes on daily basis. Behaves like a centralized shared folder like OneDrive

SVN CVS GITHUB-- most of the projects use this. VSS from MS Clear case from IBM

Microsoft Visual SourceSafe is a file-level version control system that permits many types of organizations to work on several project versions at the same time. This capability is particularly beneficial in a software development environment, where it is used in maintaining parallel code versions. However, the product can also be used to maintain files for any other type of team.

- Check in - write the changes into server.
- Check out - read the changes from the server.
- Sync - comparing the shared folder with project.
- Merge - if somebody updated your project, so merge the changes.

CVS "Concurrent Versions System" is a version control or tracking system. It maintains records of files through their development and allows retrieval of any stored version of a file, and supports production of multiple versions. CVS allows you to split the development into 2 or more parts called a trunk (MAIN) and a branch. You can create 1 or more branches. Typically a branch is used for bug fixes and trunk is used for future development. Both the trunk and branches are stored in the same repository. This allows the change from branch (i.e. bug fixes) to ultimately or periodically be merged into the main trunk ensuring that all bug fixes get rolled into next release.

**In GitHub** → New repository → Repository name → copy HTTP link

**In Eclipse** → show view → GIT repository → clone a GIT repository → paste the URL and connect

**Check In** → right click project → Team → GIT → finish

**Save in GitHub** → right click project → team → commit → Pushed.

- Do this process whenever changes in project.

Right Click - **Compare with** - this shows difference in versions.

Branch → Master "Current release" (Trunk - latest code)

→ Version 1 "If any changes" (Tag - V1.1-2/2015)

Most of time we use **Automated build tools**

**Agile software development** is a style of software development that emphasizes customer satisfaction through continuous delivery of functional software.

- Cruise Control - Jenkins - Anthill pro IBM

- If checked in the file with compilation errors then this automated tools, then it will identify complication error and Send message email to all project members.

- It will build test cases with the project, if test case fails then sends a mail to all.

- Output gives WAR, JAR and EAR (Enterprise archive - package of multiple Web applications)

**Interview: What methodology you have used as part of the project?**

**Agile process** - with a minimal documentation, all stages kickoff at the same time parallel.

- Every month we are going to deliver some models. Every 15 days to one month is called **SPRINT period**.
- What is deliverable for Sprint 1? **Sprint** is combination of creating user stories as USE case diagrams
- They use **Rally - project management tool** - maintained by project manager. **Use case diagrams**: Depicts the typical interaction between external users (actors) and the system. The emphasis is on what a system does rather than how it does it.
- **User stories** - Combination of simple tasks given to developers.
- Another is **SCRUM** - everyday meetings, yesterday what you have done, today what are you doing. handled by SCRUM master (project manager or any of the team member)

In AGILE- one iteration = 2 to 3 weeks 4 iterations will make a SPRINT

**Interview: Disadvantages in Agile model?**

- I didn't find any issues, only one which is a time conflicts in order to get clarified.... offshore teams,

**Waterfall model** - systematic development life cycle

**Interview: In which part you worked according to SDLC?**

- I was working on implementation layer as a java developer.

#### AGILE

- \* Last minute changes done NOT done
- \*
- \* Customer requirements can be modified
- \* continuous activities
- \* developed in 1990

#### WATERFALL

- NOT done
- Too risky
- NOT done
- one of one activities
- 1970

**BA** - gather requirements from clients

**SA** - creates Wireframes (UI screens) using USE case diagrams into Login, Transaction module.

**Developer/Technical Lead** – Involve in development of project **SRS or Software Requirement Specification** is a document produced at the time of requirement gathering process.

**Interview: Have you involved in requirement gathering?** It can be also seen as a process of refining requirements

- Yes we used to meet BA, SA meetings but we have not directly involved with the client. and documenting them.
- Whenever some clarifications is needed, then we setup a meeting a client directly.

**Interview: Have involved in the design of the project, if so what design you did?**

- Yes, by looking on the project documentation, we are going to decide as I attend the design meetings with architects and I also used to give some suggestions about how to achieve this functionality to fulfill this business requirements.
- Architect will decide X framework for the project, but as a developer, I used to evaluate whether this framework will suitable for the project or not. If we find any drawbacks then we discuss with the architects.

**Drawback in waterfall model:**

- If we miss any requirement, then we used to redo the entire process.

**Maven Install** -- package the project as JAR file and use that in other project as maven dependencies.

**Waterfall approach** is sequential in nature. The iterative approach is non-sequential and incremental.

This model is one of the oldest models and is widely used in government projects and in many major companies.

sys req -> S/W req -> archi and detailed design -> coding -> testing -> maintainance

# DAY 44 Spring Security

## Web Application:

### Two Parts:

- UI Web App
- Service WebApp - Java client "Service SDK" implemented using SOAP/REST web services.
- Service team has to provide the SDK to the UI team.

### Integration of UI Web APP and Service WebApp:

Adapter class is going to invoke the client class to get the data from the service layer "calling the service SDK" → Model object with service response → "DozerBeanMapper" map it to the UI object.

### Spring Security:

- It provides security for several web resources based on URL pattern. Such as secure/user/\* all must be authenticate.
- secure/admin/\* gives more roles and access.

### Web App consists of 2 pages:

- Public page: all can view like login/logout
- Private page: only authenticate users can view like transfer funds, account details.

Something is intercepting your request - **Front controller design pattern**.

- In normal Web APP - we use **dispatcher servlet**.

### Interview: Have you used LDAP server?

LDAP server - authentication server using username and password.

- When password will expire.
- Stores the user details along with user roles such as GMAIL, yahoo?
- Spring security is configured to interact with the LDAP server.

In APP, we will not store password directly in database. We will be using encrypted mechanism, we are going to use **one way hash mechanism** - there is no way to reverse engineer of the password.

### Interview: How you maintained a database in a server?

- We have configured the database in server using data source → configured as a part of spring.
- Instead of using to specify the details we use JNDI.

### In spring security.xml:

**Authentication manager:** how you would like to authenticate the user, what procedure you want to use?

```
<authentication-manager>
 <authentication-provider>
 <jdbc-user-service data-source-ref="dataSource">
 users-by-username-query=
 "select username,password,enabled
 from users where username=?"
 authorities-by-username-query=
 "select username, role from user_roles
 where username =? "
 //authorities - used to specify roles.
```

```
<intercept-url pattern="/admin**"
access="hasRole('ROLE_ADMIN')"/>
 • Matches the URL pattern
```

**Ehcache** is an open source, standards-based cache for boosting performance, offloading your database, and simplifying scalability.

- It's the most widely-used Java-based cache because it's robust, proven, and full-featured.

# DAY 45 JNDI

## Queue Browser:

- Instead of queue sender and receiver, we use queue browser to read the messages from the queue.

## JNDI - java naming directive interface

- light weight API - inside APP server → provide database/queue services with an object with JNDI name
- In java APP → get the services from server is identified using JNDI name.

WebLogic is done by Administrator not by Java developer.

Domain - folder in which it contains application details.

- WebLogic creates interface server "own server" configure few services what application requires.
- It is a commercial server - run APP according to instance server.
- Restricting some changes deployed APP in one specific server, this is mirror mechanism called **clustering**.
- One server down → it carries to other server.

## Start the server:

- http://localhost:7001/console
- username:weblogic password:shreya123

## Services:

- Production server:** we should put database credentials in code or properties file. It can be easily hacked and manipulated.
- 99% we use server side database connection management,

In Data sources in WebLogic → Create a New JDBC Data Source → JNDI Name (jndi/mysql) "all credentials must store in this object"

In Data source → targets → adminserver

Java App client → using JNDI API fetch → WebLogic jndi name

## Creating a wlfullclient.jar for JDK 1.6 client applications:

- Change directories to the server/lib directory.
- cd WL\_HOME/server/lib
- Use the following command to create wlfullclient.jar in the server/lib directory:
- java -jar wljarbuilder.jar
- You can now copy and bundle the wlfullclient.jar with client applications.
- Add the wlfullclient.jar to the client application's class path.

## In DBUtil.java:

// To establish a connection.

```
public class DBUtil {
 public static Connection getDBconnection() throws Ex
 {
 // DataSource DS = (DataSource) getService("jdbc/mysql");
 // key to be mapped
 DataSource DS = (DataSource)
 ServiceLocator.getService("jdbc/mysql");
 Connection con = DS.getConnection();
 return con;
 }
 public class ServiceLocator {
 private static Context con = null; // singleton behavior
 static{ // will execute only one time.
 Hashtable<String, String> weblogicProp = new Hashtable<String, String>();
 weblogicProp.put(Context.INITIAL_CONTEXT_FACTORY,
 "weblogic.jndi.WLInitialContextFactory"); // This
 setting identifies the factory that actually creates the Context.
 }
 }
}
```

```
weblogicProp.put(Context.PROVIDER_URL,"t3://localhost:7001");
// URL
weblogicProp.put(Context.SECURITY_PRINCIPAL,"weblogic");
// username
weblogicProp.put(Context.SECURITY_CREDENTIALS,"shreya123");
// password
con = new InitialContext(weblogicProp);
public static Object getService(String jndiName) throws Exception
 return con.lookup(jndiName);
```

## Using WebLogic JNDI to Connect a Java Client to a Single Server:

- Set up JNDI environment properties for the InitialContext.
- Establish an InitialContext with WebLogic Server.
- Use the Context to look up a named object in the WebLogic Server namespace (domain structure).
- Use the named object to get a reference for the remote object and invoke operations on the remote object.
- Close the context.
- Instead of creating this JNDI connection in all layers, am creating a serviceLocator class:

## Interview: what are the service patterns you know?

- Service Locator** - Lookup object in JNDI, Simplify the lookup process, Improve the performance
- Singleton combined with service locator to make sure only one lookup object exists.

## Interview: what is the use of JNDI and how you can get the object?

- my data source object is configured in my server and recognized with a JNDI name
- need to create initial context object to pass factory name driver class, URL, username and password "database properties"

## JMS servers in WebLogic:

- create a new JMS server → persistence "you can store messages" → target → finish

## JMS modules in WebLogic:

- create a new JMS module → admin sever → finish
- inside created JMS module → click new → you will find many resources → click connectionFactory → name → specify jndi name "jndi/jms/connectionFactory"
- Same create a template and then create a queue inside created JMS module → click new → you will find many resources → click queue → name → specify jndi name "jndi/jms/queue" → template → create a new myQueueSubdeployment → myJMS server

## In JMSproject in eclipse:

QueueSender: //Get connection factory from weblogic configured with JNDI name

```
ConnectionFactory CF = (ConnectionFactory)
ServiceLocator.getService("jndi/jms/connectionFactory")
Destination queue = (Destination)
ServiceLocator.getService("jndi/jms/queue");
MessageProducer MP = ses.createProducer(queue);
```

In monitoring: you can see the messages...



# TOP 50 SERVLETS

## 1. What is a Servlet?

A servlet is a Java technology and it is managed by a container called servlet engine. It generates dynamic content and interacts with client through Request and Response.

## 2. Why servlet is mostly used?

Servlets are mostly used because they are platform-independent Java classes and are compiled to platform-neutral byte code. Java byte code can be loaded dynamically into and run by java enabled web server.

## 3. What is called servlet container?

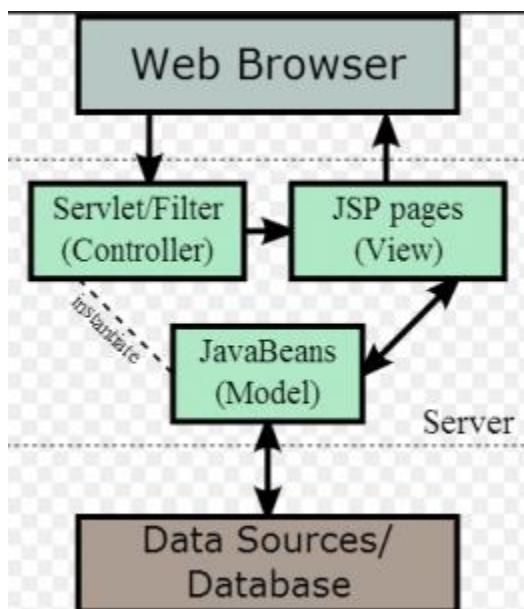
A servlet container is a part of Web server that provides network services depends on request and response are sent, MIME based requests and responses. It contains and manages servlets through their life cycle.

## 4. What is a filter?

A filter is nothing but a piece of code which can be reusable that will be transforming the content of HTTP requests, response and header information.

## 5. How can we refresh automatically when new data has entered the database?

Refresh in Client side and Server Push can be performed to refresh automatically when new data is entered into the database.



*Servlet*

## **6. What is called a session?**

A session is an object which is used by a servlet and it is used to track user interaction with a web application across multiple HTTP requests.

## **7. What is servlet mapping?**

Servlet Mapping is an association mapping between servlet and a URL pattern. This is used to map servlets with the requests.

<servlet-mapping> // mapping the servlet with URL patterns

<servlet-name>xmlServletNAME</servlet-name>

<url-pattern>/xmlServletPath</url-pattern> // change the pathname

</servlet-mapping>

Servlet context contains servlet view of Web application in which servlet will be running. By using the context,

- Log events
- Obtain URL references to resources
- Set and Store attributes

## **9. Which interface should be implemented by all servlets?**

Servlet interface should be implemented by all servlets.

## **10. What is life cycle of Servlet?**

Following is life cycle of Servlet:

- Loaded
- Initialized
- Destroy
- Unloaded

## **11. What is the difference between Servlet Request and Servlet Context when calling a Request Dispatcher?**

Relative URL can be called when Servlet Request is used and Relative URL is not used when using Servlet Context.

## **12. What are the features added in Servlet 2.5?**

Following are the features added in Servlet 2.5:

- Dependency on J2SE 5.0
- Support for annotations
- Loading the class
- Several web.xml
- Removed restrictions
- Edge case clarifications

### **13. When servlet is loaded?**

A servlet can be loaded when:

- First request is made
- Auto loading and Server starts up
- There is a single instance that answers all requests concurrently which saves memory
- Administrator manually loads.

### **14. When Servlet is unloaded?**

A servlet is unloaded when:

- Server shuts down
- Administrator manually unloads

### **15. What are the supporting protocol by HttpServlet?**

HttpServlet supports only HTTP and HTTPS protocol.

### **16. What is called Session Tracking?**

Session tracking is used to maintain a state on the series of requests from the same user for a given period of time.

### **17. Why session tracking is needed?**

Every HTTP request needs to be captured by HTTP protocol and for that, state is captured. Tracking of state is called session tracking.

### **18. What are the types of Session Tracking?**

There are following types of session tracking:

- URL rewriting
- Hidden Form Fields
- Cookies
- Secure Socket Layer (SSL)

### **19. What are the advantages of cookies?**

Cookies are used to store long term information that can be maintained without server interaction. Small and Medium size data are kept in a queue.

### **20. What is URL rewriting?**

URL rewriting is one of the methods of session tracking in which additional data is appended at the end of each URL. This additional data identifies the session.

## **21. What is servlet lazy loading?**

A servlet container which does not initialize at the start up, this is known as servlet lazy loading.

## **22. What is Servlet Chaining?**

Chaining is one of the methods where out of one servlet is given to the second servlet. This chaining can happen for any number of servlets.

## **23. What are the important functions of filters?**

Following are the important functions of Filters:

- Security check
- Modifying the request or response
- Data compression
- Logging and auditing
- Response compression

## **24. What are the functions of Servlet container?**

Following are the functions of the Servlet container:

- |                          |                                                                                                                    |
|--------------------------|--------------------------------------------------------------------------------------------------------------------|
| • Lifecycle management   | <b>JSP</b><br>* JSP is a webpage scripting language that can generate dynamic content.                             |
| • Communication support  | * JSP run slower compared to Servlet                                                                               |
| • Multithreading support | as it takes compilation time to convert into Java Servlets.<br>* It's easier to code in JSP than in Java Servlets. |
| • Declarative security   | * In MVC, jsp act as a view.                                                                                       |
| • JSP support            | * In MVC, servlet act as a controller.<br>* we can build custom tags                                               |

## **25. What is the difference between JSP and Servlets?**

- |                                                                                                                                 |                                                                                                                   |
|---------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| JSP supports HTTP protocol which mainly used for presentation. But a servlet can support any protocol like HTTP, FTP, SMTP etc. | <b>Servlets</b><br>* Servlets are Java programs that are already compiled which also creates dynamic web content. |
|                                                                                                                                 | * Servlets run faster compared to JSP.                                                                            |
|                                                                                                                                 | * Its little much code to write here.                                                                             |
|                                                                                                                                 | * In MVC, servlet act as a controller.                                                                            |
|                                                                                                                                 | * no such custom tag facility in servlets.                                                                        |

## **26. What are all the ways for session tracking?**

Following are the ways for session tracking:

- Cookies
- URL rewriting
- HttpSession
- Hidden form fields

## **27. What is called Scriptlet?**

A scriptlet contains any language statements, variables, expressions that can be valid in the page scripting language. Scriptlet is a part of generated servlet service method.

## **28. What is the difference between Server and Container?**

A server can provide service to the client and it contains one or more containers such as EJBs, Servlet, JSP containers. Containers hold set of objects.

## **29. Can we refresh servlet in client and server side automatically?**

On the client side, Meta http is used for refresh and server push is used for server side refresh.

## **30. What is the difference between ServletConfig and ServletContext?**

ServletConfig provides information about configuration of a servlet which is defined inside the web.xml file and it is a specific object for each servlet.

ServletContext is an application specific object and it is shared by all servlet. It belongs to one application in one JVM.

## **31. What is Pure Servlet?**

Pure servlet is servlet which is used to create java objects that can be implemented from javax.servlet.Servlet interface.

## **32. What is the difference between Servlets and applets?**

Servlets are used for server side config and it keeps on server. But, Applets are used for client side coding and it runs on client browsers.

## **33. What is Generic Servlet class?**

Generic servlet is the super class of all servlets. This class is extended by all other classes and it is protocol independent.

## **34. What is Java Servlet?**

Java servlet is used to provide secure access to the web based date. This can extend functionality present in the web servers. As it is platform and server independent, it is used for many purposes.

## **35. What is StringTokenizer?**

A StringTokenizer is used to break the string into tokens and the token value is passed as an argument in the constructor.

## **36. What is HttpServlet and how it is different from GenericServlet?**

HttpServlet extends from GenericServlet and inherits the properties of GenericServlet. HttpServlet defines a HTTP protocol servlet while GenericServlet defines a generic, protocol-independent servlet.

### **37. How to get the current HttpSession object?**

GetSession method is used to get the current HttpSession object on HttpServletRequest.

### **38. What do you mean by Default initialization in Java Servlet?**

This is one of the servlet initialization and it is initialized when it is called for the first time.

### **39. What is Servlet Invoker?**

Servlet Invoker allows web application to dynamically register new servlet definitions with the servlet tag in the /WEB-INF/web.xml.

### **40. What is called Servlet mapping?**

Servlet mapping maps URL patterns with the servlets. If there is a request from the client, servlet container decides on which application it needs to map.

### **41. What are all the protocols supported by HttpServlet?**

HttpServlet supports HTTP and HTTPS protocol.

### **42. Which exception is thrown if servlet is not initialized properly?**

Servlet Exception or Unavailable Exception is thrown if servlet is not initialized properly.

### **43. Who is responsible for writing a constructor?**

Container is responsible for writing constructor without arguments in servlet.

### **44. What are all the advantages of Servlet over CGI?**

Following are the advantages of Servlet over CGI:

- Cannot be run in an individual process.
- Servlet stays in the memory while requests. For every CGI request, you must load and start a CGI program.
- web.xml conveniences

Common Gateway Interface (CGI) is a standard method used to generate dynamic content on Web pages and Web applications

### **45. What are the different mode that servlets can be used?**

Following are the modes that servlets can be used:

- Filter chains can be used to collect servlets together
- Support HTTP protocol
- Used for CGI based applications
- Dynamic generation of servlets

#### **46. What are the uses of servlets?**

Servlets are used to process and store data submitted by HTML form, dynamic content, handle multiple request concurrently and manage state information on top of stateless HTTP.

#### **47. Whether we can get deadlock situation in Servlets?**

Yes, it can be achieved by writing doGet method in doPost method and writing doPost method in doGet method.

#### **48. What is the default HTTP method in the servlet?**

Default method is GET method for HTTPservlet.

#### **49. Whether thread can be used in Servlets?**

Yes, Single thread can be used in servlets.

#### **50. What exception should be thrown when servlet is not properly initialized?**

Servlet exception or an Unavailable exception is thrown when it is not properly initialized.

An average recruiter has 6 seconds to check your resume... will you be noticed?

[Check Premium Resumes](#)



## Top 50 SQL Question & Answers

### 1. What is DBMS?

A Database Management System (DBMS) is a program that controls creation, maintenance and use of a database. DBMS can be termed as File Manager that manages data in a database rather than saving it in file systems.

### 2. What is RDBMS?

RDBMS stands for Relational Database Management System. RDBMS store the data into the collection of tables, which is related by common fields between the columns of the table. It also provides relational operators to manipulate the data stored into the tables.

**Example:** SQL Server.

### 3. What is SQL?

SQL stands for Structured Query Language , and it is used to communicate with the Database. This is a standard language used to perform tasks such as retrieval, updation, insertion and deletion of data from a database.

Standard SQL Commands are Select.

### 4. What is a Database?

Database is nothing but an organized form of data for easy access, storing, retrieval and managing of data. This is also known as structured form of data which can be accessed in many ways.

Example: School Management Database, Bank Management Database.

### 5. What are tables and Fields?

A table is a set of data that are organized in a model with Columns and Rows. Columns can be categorized as vertical, and Rows are horizontal. A table has specified number of column called

fields but can have any number of rows which is called record.

Example:.

Table: Employee.

Field: Emp ID, Emp Name, Date of Birth.

Data: 201456, David, 11/15/1960.

## 6. What is a primary key?

A primary key is a combination of fields which uniquely specify a row. This is a special kind of unique key, and it has implicit NOT NULL constraint. It means, Primary key values cannot be NULL.

Displaying the column data which eliminates the duplicate of rows ... just print unique rows which doesn't repeat twice

## 7. What is a unique key? or more....

SELECT DISTINCT city FROM customers

A Unique key constraint uniquely identified each record in the database. This provides uniqueness for the column or set of columns.

A Primary key constraint has automatic unique constraint defined on it. But not, in the case of Unique Key.

There can be many unique constraint defined per table, but only one Primary key constraint defined per table.

CREATE TABLE Orders

(

## 8. What is a foreign key?

O\_Id int NOT NULL, OrderNo int NOT NULL,  
P\_Id int, PRIMARY KEY (O\_Id), FOREIGN KEY

A foreign key is one table which can be related to the primary key of another table. Relationship needs to be created between two tables by referencing foreign key with the primary key of another table.

## 9. What is a join?

Returns rows when there is atleast one match in both tables

This is a keyword used to query data from more tables based on the relationship between the fields of the tables. Keys play a major role when JOINs are used.

## 10. What are the types of join and explain each?

There are various types of join which can be used to retrieve data and it depends on the relationship between tables.

### Inner join.

All rows common between left and right tables

Inner join return rows when there is at least one match of rows between the tables.

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
INNER JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
```

---

```
SELECT Orders.OrderID, Employees.FirstName
```

```
FROM Orders
```

**Right Join.**

```
RIGHT JOIN Employees
```

```
ON Orders.EmployeeID=Employees.EmployeeID
```

Right join return rows which are common between the tables and all rows of Right hand side table. Simply, it returns all the rows from the right hand side table even though there are no matches in the left hand side table. **Right outer:** Rows of right without matching rows in left table

**Left Join.**

Left join return rows which are common between the tables and all rows of Left hand side table. Simply, it returns all the rows from Left hand side table even though there are no matches in the Right hand side table. **Left outer:** Rows of left without matching rows in right table

```
SELECT Customers.CustomerName, Orders.OrderID
```

```
FROM Customers
```

```
FULL OUTER JOIN Orders
```

```
ON Customers.CustomerID=Orders.CustomerID
```

Full join return rows when there are matching rows in any one of the tables. This means, it returns all the rows from the left hand side table and all the rows from the right hand side table.

**All rows from both left and right tables except the common/matching ones**

## 11. What is normalization?

Normalization is the process of minimizing redundancy and dependency by organizing fields and table of a database. The main aim of Normalization is to add, delete or modify field that can be made in a single table.

**Data Definition Language (DDL)** - create, alter, rename, drop, truncate

**Data Manipulation Language (DML)** - select, insert, update, delete (CRUD)

**Data Control Language (DCL)** - grant, revoke

## 12. What is Denormalization.

**Transaction Control Language (TCL)** - commit, rollback, savepoint  
database managers use to increase the performance of a database infrastructure

DeNormalization is a technique used to access the data from higher to lower normal forms of database. It is also process of introducing redundancy into a table by incorporating data from the related tables.

```
CREATE TABLE denormalized_settings (
 user_id int NOT NULL,
 setting1 varchar(255) NULL,
 setting2 varchar(255) NULL,
 setting3 varchar(255) NULL,
 setting4 varchar(255) NULL
```

The normal forms can be divided into 5 forms, and they are explained below -.

### First Normal Form (1NF):

```
INSERT INTO denormalized_settings (user_id, setting1,
setting2, setting3, setting4) VALUES
(1, 'hi', '0', '1', '0'),
(2, '0', '1', NULL, '1'),
(3, '1', '0', '1', NULL);
```

This should remove all the duplicate columns from the table. Creation of tables for the related data and identification of unique columns.

### Second Normal Form (2NF):..

Meeting all requirements of the first normal form. Placing the subsets of data in separate tables and Creation of relationships between the tables using primary keys.

### Third Normal Form (3NF):..

This should meet all requirements of 2NF. Removing the columns which are not dependent on

primary key constraints.

#### Fourth Normal Form (3NF):..

Meeting all the requirements of third normal form and it should not have multi-valued dependencies.

	View	Materialized View
	Runs the query each time it is accessed	Runs the query only once and updates periodically
<b>14. What is a View?</b>	Gets latest data but performance depends on well-formed query	Fast but may not refresh on time to get latest data

A view is a virtual table which consists of a subset of data contained in a table. Views are not virtually present, and it takes less space to store. View can have data of one or more tables combined, and it is depending on the relationship.

Performance tuning can be done in the following ways :

<b>15. What is an Index?</b>	- Too much normalization is bad, denormalize where required optimize index usage	- Too much of indexing is bad, reduce no.of.cols that make up the composite key of table spaces, have separate space for BLOBS and CLOBs	- Proper partitioning - Use stored procedures
------------------------------	-------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------

An index is performance tuning method of allowing faster retrieval of records from the table. An index creates an entry for each value and it will be faster to retrieve data.

#### 16. What are all the different types of indexes?

There are three types of indexes -.

##### Unique Index.

This indexing does not allow the field to have duplicate values if the column is unique indexed. Unique index can be applied automatically when primary key is defined.

##### Clustered Index.      Telephone directory

This type of index reorders the physical order of the table and search based on the key values. Each table can have only one clustered index.      Sorting of the indexes sorts the data

##### NonClustered Index.      Index at the back of the book

NonClustered Index does not alter the physical order of the table and maintains logical order of data. Each table can have 999 nonclustered indexes.      Sorting of the indexes does not sort the data

#### 17. What is a Cursor?

A database Cursor is a control which enables traversal over the rows or records in the table. This can be viewed as a pointer to one row in a set of rows. Cursor is very much useful for traversing such as retrieval, addition and removal of database records.

#### 18. What is a relationship and what are they?

Database Relationship is defined as the connection between the tables in a database. There

are various data basing relationships, and they are as follows:.

- One to One Relationship.
- One to Many Relationship.
- Many to One Relationship.
- Self-Referencing Relationship.

## 19. What is a query?

A DB query is a code written in order to get the information back from the database. Query can be designed in such a way that it matched with our expectation of the result set. Simply, a question to the Database.

## 20. What is subquery?

A subquery is a query within another query. The outer query is called as main query, and inner query is called subquery. SubQuery is always executed first, and the result of subquery is passed on to the main query.

## 21. What are the types of subquery?

There are two types of subquery – Correlated and Non-Correlated.

A correlated subquery cannot be considered as independent query, but it can refer the column in a table listed in the FROM the list of the main query.

A Non-Correlated sub query can be considered as independent query and the output of subquery are substituted in the main query.

## 22. What is a stored procedure?

Stored Procedure is a function consists of many SQL statement to access the database system. Several SQL statements are consolidated into a stored procedure and execute them whenever and wherever required.

## 23. What is a trigger?

A DB trigger is a code or programs that automatically execute with response to some event on a table or view in a database. Mainly, trigger helps to maintain the integrity of the database.

Example: When a new student is added to the student database, new records should be created in the related tables like Exam, Score and Attendance tables.

## 24. What is the difference between DELETE and TRUNCATE commands?

DELETE command is used to remove rows from the table, and WHERE clause can be used for

conditional set of parameters. Commit and Rollback can be performed after delete statement.

TRUNCATE removes all rows from the table. Truncate operation cannot be rolled back. autocommit trims the table....

## 25. What are local and global variables and their differences?

Local variables are the variables which can be used or exist inside the function. They are not known to the other functions and those variables cannot be referred or used. Variables can be created whenever that function is called.

Global variables are the variables which can be used or exist throughout the program. Same variable declared in global cannot be used in functions. Global variables cannot be created whenever that function is called.

## 26. What is a constraint?

Constraint can be used to specify the limit on the data type of table. Constraint can be specified while creating or altering the table statement. Sample of constraint are.

- NOT NULL.
- CHECK.
- DEFAULT.
- UNIQUE.
- PRIMARY KEY.
- FOREIGN KEY.

## 27. What is data Integrity?

Data Integrity defines the accuracy and consistency of data stored in a database. It can also define integrity constraints to enforce business rules on the data when it is entered into the application or database.

## 28. What is Auto Increment?

Auto increment keyword allows the user to create a unique number to be generated when a new record is inserted into the table. AUTO INCREMENT keyword can be used in Oracle and IDENTITY keyword can be used in SQL SERVER.

Mostly this keyword can be used whenever PRIMARY KEY is used.

## 29. What is the difference between Cluster and Non-Cluster Index?

Clustered index is used for easy retrieval of data from the database by altering the way that the records are stored. Database sorts out rows by the column which is set to be clustered index.

A nonclustered index does not alter the way it was stored but creates a complete separate

object within the table. It point back to the original table rows after searching.

### 30. What is Datawarehouse?

Datawarehouse is a central repository of data from multiple sources of information. Those data are consolidated, transformed and made available for the mining and online processing. Warehouse data have a subset of data called Data Marts.

### 31. What is Self-Join?

Self-join is set to be query used to compare to itself. This is used to compare values in a column with other values in the same column in the same table. ALIAS ES can be used for the same table comparison.

### 32. What is Cross-Join?

Cross join defines as Cartesian product where number of rows in the first table multiplied by number of rows in the second table. If suppose, WHERE clause is used in cross join then the query will work like an INNER JOIN.

### 33. What is user defined functions?

User defined functions are the functions written to use that logic whenever required. It is not necessary to write the same logic several times. Instead, function can be called or executed whenever needed.

### 34. What are all types of user defined functions?

Three types of user defined functions are.

- Scalar Functions.
- Inline Table valued functions.
- Multi statement valued functions.

Scalar returns unit, variant defined the return clause. Other two types return table as a return.

### 35. What is collation?

Collation is defined as set of rules that determine how character data can be sorted and compared. This can be used to compare A and, other language characters and also depends on the width of the characters.

ASCII value can be used to compare these character data.

## 36. What are all different types of collation sensitivity?

Following are different types of collation sensitivity -

- Case Sensitivity – A and a and B and b.
- Accent Sensitivity.
- Kana Sensitivity – Japanese Kana characters.
- Width Sensitivity – Single byte character and double byte character.

## 37. Advantages and Disadvantages of Stored Procedure?

Stored procedure can be used as a modular programming – means create once, store and call for several times whenever required. This supports faster execution instead of executing multiple queries. This reduces network traffic and provides better security to the data.

Disadvantage is that it can be executed only in the Database and utilizes more memory in the database server.

## 38. What is Online Transaction Processing (OLTP)?

Online Transaction Processing or OLTP manages transaction based applications which can be used for data entry and easy retrieval processing of data. This processing makes like easier on simplicity and efficiency. It is faster, more accurate results and expenses with respect to OTLP.

Example – Bank Transactions on a daily basis.

## 39. What is CLAUSE?

SQL clause is defined to limit the result set by providing condition to the query. This usually filters some rows from the whole set of records.

Example – Query that has WHERE condition

Query that has HAVING condition.

## 40. What is recursive stored procedure?

A stored procedure which calls by itself until it reaches some boundary condition. This recursive function or procedure helps programmers to use the same set of code any number of times.

## 41. What is Union, minus and Interact commands?

UNION operator is used to combine the results of two tables, and it eliminates duplicate rows from the tables.

MINUS operator is used to return rows from the first query but not from the second query.

Matching records of first and second query and other rows from the first query will be displayed as a result set.

INTERSECT operator is used to **return rows returned by both the queries.**

#### 42. What is an ALIAS command?

ALIAS name can be given to a table or column. This alias name can be referred in **WHERE clause to identify the table or column.**

Example-.

[crayon-54f5a4b9bf1f7681709344/]

Here, **st** refers to alias name for student table and **Ex** refers to alias name for exam table.

#### 43. What is the difference between TRUNCATE and DROP statements?

TRUNCATE **removes all the rows from the table**, and **it cannot be rolled back**. DROP command **removes a table from the database** and **operation cannot be rolled back**.

#### 44. What are aggregate and scalar functions?

Aggregate functions are **used to evaluate mathematical calculation and return single values**. This can be **calculated from the columns in a table**. Scalar functions **return a single value based on the input value**.

Example -.

Aggregate – **max()**, **count** - Calculated with respect to numeric.

Scalar – **UCASE()**, **NOW()** – Calculated with respect to strings.

#### 45. How can you create an empty table from an existing table?

Example will be -.

Select \* into studentcopy from student where 1=2

[crayon-54f5a4b9bf200384312324/]

Here, we are **copying student table to another table** with the same structure with no rows copied.

#### 46. How to fetch common records from two tables?

Select studentID from student. **INTERSECT** Select StudentID from Exam  
Common records result set can be achieved by -.

[crayon-54f5a4b9bf203971586569/]

## 47. How to fetch alternate records from a table?

Records can be fetched for both Odd and Even row numbers -.

To display even numbers.     Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=0  
[crayon-54f5a4b9bf206431213812/]

To display odd numbers-.     Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=1  
[crayon-54f5a4b9bf208315278133/]

from (Select rowno, studentId from student) where mod(rowno,2)=1.[/sql]

## 48. How to select unique records from a table?

Select unique records from a table by using DISTINCT keyword.  
[crayon-54f5a4b9bf20b400339250/]

Select DISTINCT StudentID, StudentName from Student.

## 49. What is the command used to fetch first 5 characters of the string?

There are many ways to fetch first 5 characters of the string -.

[crayon-54f5a4b9bf20d110291033/]

[crayon-54f5a4b9bf210932945644/]

    Select SUBSTRING(StudentName,1,5) as studentname from student

    MySQL

    Select RIGHT(Studentname,5) as studentname from student

## 50. Which operator is used in query for pattern matching?

LIKE operator is used for pattern matching, and it can be used as -.

1. % - Matches zero or more characters.
2. \_(Underscore) – Matching exactly one character.

Example -.

[crayon-54f5a4b9bf212312458368/]

[crayon-54f5a4b9bf215856700698/]

    Select \* from Student where studentname like 'a%'

    Select \* from Student where studentname like 'ami\_'

---

Guru99 provides [FREE ONLINE TUTORIAL](#) on Various courses like

[PHP](#)

[Java](#)

[Linux](#)

[Apache](#)

[Perl](#)

[SQL](#)

[VB Script](#)

[JavaScript](#)

[Accounting](#)

[Ethical Hacking](#)

[Cloud Computing](#)

[Jmeter](#)

[Manual Testing](#)

[QTP](#)

[Selenium](#)

[Test Management](#)

[Load Runner](#)

[Quality Center](#)

[Mobile Testing](#)

[Live Selenium Project](#)

[Enterprise Testing](#)

[Live Testing Project](#)

[Sap & All Modules](#)



Copyrighted Material

# TOP 50 STRUTS

Open source framework built on MVC architecture for developing web applications using Java EE.

## Q1. What are the components of Struts Framework?

Ans: Struts framework is comprised of following components:

1. Java Servlets
2. JSP (Java Server Pages)
3. Custom Tags
4. Message Resources

**Struts Life Cycle : when a req comes from a jsp or html,**  
- Retrieve or create form bean  
- Store form bean in appropriate scope  
- Reset properties/attributes  
- Populate properties  
- Validate properties  
- Pass form bean to action class

Model: Java beans, EJB  
View: HTML, JSP  
Controller: ActionServlet

## Q2. What's the role of a handler in MVC based applications?

Ans: It's the job of handlers to transfer the requests to appropriate models as they are bound to the model layer of MVC architecture. Handlers **use mapping information from configuration files for request transfer.**

## Q3. What's the flow of requests in Struts based applications?

Ans: Struts based applications use MVC design pattern. The flow of requests is as follows:

- User interacts with **View** by clicking any link or by submitting any form.
- Upon user's interaction, the request is passed towards the **controller**.
- **Controller** is responsible for passing the request to appropriate **action**.
- **Action** is responsible for calling a function in **Model** which has all business logic implemented.
- Response from the **model** layer is received back by the **action** which then passes it towards the **view** where user is able to see the response.

## Q4. Which file is used by controller to get mapping information for request routing?

Ans: Controller uses a **configuration file "struts-config.xml file** to get all mapping information to decide which **action** to use for routing of user's request.

## Q5. What's the role of Action Class in Struts?

# Struts

Adapter between request contents and logic that need to be executed.  
**execute()- business logic resides here**  
**execute(ActionMapping, ActionForm, Request, Response)**

Ans: In Struts, Action Class **acts as a controller** and performs following key tasks:

- After receiving user request, it processes the user's request.
- Uses appropriate model and pulls data from model (if required).

- Selects proper view to show the response to the user.

## Q6. How an actionForm bean is created?

### Surrogate

Ans: actionForm bean is created by extending the class org.apache.struts.action.ActionForm

In the following example we have created an actionForm bean with the name 'testForm':

```

1 import javax.servlet.http.HttpServletRequest;
2 import org.apache.struts.action.*;
3 public class testForm extends ActionForm {
4 private String Id=null;
5 private String State=null;
6 public void setId(String id){
7 this.Id=id;
8 }
9 public String getId(){
10 return this.Id;
11 }
12 public void setState(String state){
13 this.State=state;
14 }
15 public String getState(){
16 return this.State;
17 }

```

Java beans representing form inputs containing request parameters from the view, referencing the action bean.

validate() – used to validate the input parameters, called before form bean is handed over to the Action class, it returns ActionErrors object.

ActionErrors validate(ActionMapping, HttpServletRequest)

## Q7. What are the two types of validations supported by Validator FrameWork?

Ans: Validator Framework is used for form data validation. This framework provides two types of validations:

1. Client Side validation on user's browser
2. Server side validation

## Q8. What are the steps of Struts Installation?

Ans: In order to use Struts framework, we only need to add Struts.Jar file in our development environment.

Once jar file is available in the CLASSPATH, we can use the framework and develop Strut based applications.

## Q9. How client side validation is enabled on a JSP form?

Ans: In order to enable client side validation in Struts, first we need to enable validator plug-in in struts-config.xml file. This is done by adding following configuration entries in this file:

```

1 <!-- Validator plugin -->
2 <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
3 <set-property
4 property="pathnames"
5 value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>
6 </plug-in>

```

Then Validation rules are defined in validation.xml file. If a form contains email field and we want to enable client side validation for this field, following code is added in validation.xml file:

```
1 <form name="testForm">
2 <field property="email"
3 depends="required">
4 <arg key="testForm.email"/>
5 </field>
6 </form>
```

## **Q10. How action-mapping tag is used for request forwarding in Struts configuration file?**

Ans: In **Struts configuration file (struts-config.xml)**, forwarding options are defined under action-mapping tag.

In the following example, when a user will click on the hyperlink **test.do**, request will be forwarded to **/pages/testing.jsp** using following configurations from struts-config.xml file:

```
1 <action path="/test" forward="/pages/testing.jsp">
```

This forwarding will take place when user will click on following hyperlink on the jsp page:

```
1 <html:link page="/test.do</strong">>Controller Example</html:link>
```

## **Q11. How duplicate form submission can be controlled in Struts?**

Ans: In Struts, action class provides two important methods which can be used to avoid duplicate form submissions.

saveToken() method of action class generates a unique token and saves it in the user's session. isTokenValid() method is used then used to check uniqueness of tokens.

## **Q12. In Struts, how can we access Java beans and their properties?**

Ans: **Bean Tag Library** is a Struts library which can be used for accessing Java beans.

## **Q13. Which configuration file is used for storing JSP configuration information in Struts?**

Ans: For JSP configuration details, **Web.xml file** is used.

## **Q14. What's the purpose of Execute method of action class?**

Ans: Execute method of action class is **responsible for execution of business logic**. If any processing is required on the user's request, it's performed in this method. This method returns **actionForward object** which routes the application to appropriate page.

In the following example, execute method will return an object of **actionForward** defined in struts-config.xml with the name "exampleAction":

```

1 import javax.servlet.http.HttpServletRequest;
2 import javax.servlet.http.HttpServletResponse;
3
4 import org.apache.struts.action.Action;
5 import org.apache.struts.action.ActionForm;
6 import org.apache.struts.action.ActionForward;
7 import org.apache.struts.action.ActionMapping;
8
9 public class actionExample extends Action
10 {
11 public ActionForward execute(
12 ActionMapping mapping,
13 ActionForm form,
14 HttpServletRequest request,
15 HttpServletResponse response) throws Exception{
16 return mapping.findForward("exampleAction");
17 }
18 }
```

Contains all maps between, views to other views, and views to action classes

## **Q15. What's the difference between validation.xml and validator-rules.xml files in Struts Validation framework?**

Ans: In Validation.xml, we define validation rules for any specific Java bean while in validator-rules.xml file, standard and generic validation rules are defined.

## **Q16. How can we display all validation errors to user on JSP page?**

Ans: To display all validation errors based on the validation rules defined in validation.xml file, we use <html:errors /> tag in our JSP file.

## **Q17. What's declarative exception handling in Struts?**

Ans: When logic for exception handling is defined in struts-config.xml or within the action tag, it's known as declarative exception handling in Struts.

In the following example, we have defined exception in struts-config.xml file for NullPointerException:

```

1 <global-exceptions>
2
3 <exception key="test.key"
4
5 Type="java.lang.NullPointerException"
6
7 Path="/WEB-INF/errors/error_page.jsp"
8
9 </global-exceptions>
```

## **Q18. What's DynaActionForm?**

Ans: DynaActionForm is a special type of actionForm class (sub-class of ActionForm Class) that's used for dynamically creating form beans. It uses configuration files for form bean creation.

## **Q19. What configuration changes are required to use Tiles in Struts?**

Ans: To **create reusable components with Tiles framework**, we need to **add following plugin definition code in struts-config.xml file:**

```
1 <plug-in className="org.apache.struts.tiles.TilesPlugin" >
2
3 <set-property property="definitions-config" value="/WEB-INF/tiles-defs.xml" />
4
5 <set-property property="moduleAware" value="true" />
6
7 </plug-in>
```

## **Q20. What's the difference between Jakarta Struts and Apache Struts? Which one is better to use?**

Ans: Both are same and there is no difference between them.

## **Q21. What's the use of Struts.xml configuration file?**

Ans: Struts.xml file is **one the key configuration files of Struts framework which is used to define mapping between URL and action**. When a user's request is received by the controller, controller uses mapping information from this file to select appropriate action class.

## **Q22. How tag libraries are defined in Struts?**

Ans: **Tag libraries are defined in the configuration file (web.xml) inside <taglib> tag as follows:**

```
1 <taglib>
2
3 <taglib-uri>/WEB-INF/struts-bean.tld</taglib-uri>
4
5 <taglib-location>/WEB-INF/struts-bean.tld</taglib-location>
6
7 </taglib>
```

## **Q23. What's the significance of logic tags in Struts?**

Ans: Use of logic tags in Struts **helps in writing a clean and efficient code at presentation layer without use of scriptlets**.

## **Q24. What are the two scope types for formbeans?**

Ans: 1. Request Scope: Formbean values are available in the current request only

2. Session Scope: Formbean values are available for all requests in the current session.

## **Q25. How can we group related actions in one group in Struts?**

Ans: To group multiple related actions in one group, we can use **DispatcherAction class**.

## **Q26. When should we use SwitchAction?**

Ans: The best scenario to use SwitchAction class is when we have a modular application with multiple modules working separately. Using SwitchAction class we can switch from a resource in one module to another resource in some different module of the application.

## **Q27. What are the benefits of Struts framework?**

Ans: Struts is **based on MVC** and hence there is a **good separation of different layers in Struts** which makes **Struts applications development and customization easy**. Use of different configuration files makes Struts applications easily configurable. Also, **Struts is open source and hence, cost effective**.

## **Q28. What steps are required to for an application migration from Struts1 to Struts2?**

Ans: Following Steps are required for Struts1 to Struts2 migration:

1. Move Struts1 actionForm to Struts2 POJO.
2. Convert Struts1 configuration file (struts-config.xml) to Struts2 configuration file (struts.xml)

## **Q29. How properties of a form are validated in Struts?**

Ans: For validation of populated properties, **validate()** method of ActionForm class is used before handling the control of formbean to Action class.

## **Q30. What's the use of reset method of ActionForm class?**

Ans: reset method of actionForm class is **used to clear the values of a form before initiation of a new request**.

## **Q31. What are disadvantages of Struts?**

Ans: Although Struts have large number of advantages associated, it also requires bigger learning curve and also reduces transparency in the development process.

Struts also **lack proper documentation and for many of its components, users are unable to get proper online resources for help**.

## **Q32. What's the use of resourcebundle.properties file in Struts Validation framework?**

Ans: resourcebundle.properties file is used to define specific error messages in key value pairs for any possible errors that may occur in the code.

This approach helps to keep the code clean as developer doesn't need to embed all error messages inside code.

### **Q33. Can I have html form property without associated getter and setter formbean methods?**

Ans: For each html form property, getter and setter methods in the formbean must be defined otherwise application results in an error.

### **Q34. How many servlet controllers are used in a Struts Application?**

Ans: Struts framework works on the concept of centralized control approach and the whole application is controlled by a single servlet controller. Hence, we require only one servlet controller in a servlet application.

### **Q35. For a single Struts application, can we have multiple struts-config.xml files?**

Ans: We can have any number of Struts-config.xml files for a single application.

We need following configurations for this:

```
1 <servlet>
2
3 <servlet-name>action</servlet-name>
4
5 <servlet-class>
6
7 org.apache.struts.action.ActionServlet
8
9 </servlet-class>
10
11 <init-param>
12
13 <param-name>config</param-name>
14
15 <param-value>
16
17 /WEB-INF/struts-config.xml
18
19 /WEB-INF/struts-config_user.xml
20
21 /WEB-INF/struts-config_admin.xml
22
23 </param-value>
24
25 </init-param>
26
27
28
29
```

30

31 </servlet>

### **Q36. Which model components are supported by Struts?**

Ans: Struts support all types of models including Java beans, EJB, CORBA. However, Struts doesn't have any in-built support for any specific model and it's the developer's choice to opt for any model.

### **Q37. When it's useful to use IncludeAction?**

Ans: IncludeAction is action class provided by Struts which is useful when an integration is required between Struts and Servlet based application.

### **Q38. Is Struts thread safe?**

Ans: Yes Struts are thread safe. In Struts, a new servlet object is not required to handle each request; rather a new thread of action class object is used for each new request.

### **Q39. What configuration changes are required to use resource files in Struts?**

Ans: Resource files (.properties files) can be used in Struts by adding following configuration entry in struts-config.xml file:

```
<message-resources parameter="com.login.struts.ApplicationResources"/>
```

### **Q40. How nested beans can be used in Struts applications?**

Ans: Struts provide a separate tag library (Nested Tag Library) for this purpose. Using this library, we can nest the beans in any Struts based application.

### **Q41. What are the Core classes of Struts Framework?**

Ans: Following are the core classes provided by Struts Framework:

- Action Class
- ActionForm Class
- ActionMapping Class
- ActionForward Class
- ActionServlet Class

### **Q42. Can we handle exceptions in Struts programmatically?**

Ans: Yes we can handle exceptions in Struts programmatically by using try, catch blocks in the code.

```
1 try {
2
3 // Struts code
4
5 }
6
7 Catch (Exception e) {
8
9 // exception handling code
10
11 }
```

#### **Q43. Is Struts Framework part of J2EE?**

Ans: Although Struts framework is based on J2EE technologies like JSP, Java Beans, Servlets etc but it's not a part of J2EE standards.

#### **Q44. How action mapping is configured in Struts?**

Ans: Action mappings are configured in the configuration file struts-config.xml under the tag <action-mapping> as follows:

```
1 <pre><action-mappings>
2 <action path="/login"
3 type="login.loginAction"
4 name="loginForm"
5 input="/login.jsp"
6 scope="request"
7 validate="true">
8 <forward name="success" path="/index.jsp"/>
9 <forward name="failure" path="/login_error.jsp"/>
10 </action>
11 </action-mappings>
```

#### **Q45. When should be opt for Struts Framework?**

Ans: Struts should be used when any or some of the following conditions are true:

- A highly robust enterprise level application development is required.
- A reusable, highly configurable application is required.
- A loosely coupled, MVC based application is required with clear segregation of different layers.

#### **Q46. Why ActionServlet is singleton in Struts?**

Ans: In Struts framework, ActionServlet acts as a controller and all the requests made by users are controlled by this controller. ActionServlet is based on singleton design pattern as only one object needs to be created for this controller class. Multiple threads are created later for each user request.

#### **Q47. What are the steps required for setting up validator framework in Struts?**

Ans: Following Steps are required to setup validator framework in Struts: – **Wrong Spelling**

1. In WEB-INF directory place validator-rules.xml and validation.xml files.
2. Enable validation plugin in struts-config.xml files by adding following:

```
1 <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
2 <set-property property="pathnames" value="/WEB-INF/validator-rules.xml,
3 /WEB-INF/validation.xml"/>
4 </plug-in>
```

#### **Q48. Which technologies can be used at View Layer in Struts?**

Ans: In Struts, we can use any of the following technologies in view layer:

- **JSP**
- **HTML**
- **XML/XSLT**
- WML Files
- Velocity Templates
- **Servlets**

#### **Q49. What are the conditions for actionForm to work correctly?**

Ans: ActionForm must fulfill following conditions to work correctly:

- It must have a no argument constructor.
- It should have public getter and setter methods for all its properties.

#### **Q50. Which library is provided by Struts for form elements like check boxes, text boxes etc?**

Ans: Struts provide **HTML Tags library** which can be used for adding form elements like text fields, text boxes, radio buttons etc.

# TOP 50 WEB SERVICES

## 1) Define Web Service?

A web service is a kind of software that is accessible on the Internet. It makes use of the XML messaging system and offers an easy to understand, interface for the end users.

## 2) What is new in this field for past few years?

The initiation of XML in this field is the advancement that provides web service a single language to communicate in between the RPCs, web services and their directories.

## 3) Give me an example of real web service?

One example of web services is IBM Web Services browser. You can get it from IBM Alphaworks site. This browser shows various demos related to web services. Basically web services can be used with the help of SOAP, WSDL, and UDDI. All these, provide a plug-and-play interface for using web services such as stock-quote service, a traffic-report service, weather service etc.

**Service transport layer:** Responsible for transporting messages between applications using HTTP, FTP, and SMTP

**XML messaging layer:** Responsible for encoding messages into common XML format so that both provider and consumer can understand

## 4) How you define web service protocol stack?

**Service discovery layer:** Responsible for centralized distributed object lookup providing publish/subscribe using UDDI

It is basically set of various protocols that can be used to explore and execute web services. The entire stack has four layers i.e. Service Transport, XML Messaging, Service Description and Service Discovery.

**Service description layer:** Responsible for describing the public interface for the service using WSDL

## 5) Can you define each of these layers of protocol stack?

The Service Transport layer transfer messages between different applications, such as HTTP, SMTP, FTP, and Blocks Extensible Exchange Protocol (BEEP). The XML Messaging layer encodes messages in XML format so that messages can be understood at each end, such as XML-RPC and SOAP. The Service Description layer describes the user interface to a web service, such as WSDL. The Service Discovery layer centralizes services to a common registry and offer simple publish functionality, such as UDDI.

## 6) Define XML – RPC?

It is a protocol that makes use of XML messages to do Remote Procedure Calls.

## 7) Define SOAP?

SOAP is an XML based protocol to transfer between computers.

## **8) Define WSDL?**

It means **Web Services Description Language**. It is basically the service description layer in the web service protocol stack. The **Service Description layer** describes the user interface to a web service.

## **9) What kind of security is needed for web services?**



*Web Services*

The security level for web services should be more than that of what we say **Secure Socket Layer (SSL)**. This level of security can be only achieved from Entrust Secure Transaction Platform. Web services need this level of security to ensure reliable transactions and secure confidential information .

## **10) Do you have any idea about foundation security services?**

As implies from its name, these services are the foundation or **basics of integration, authentication, authorization, digital signatures and encryption processes**.

## **11) Define Entrust Identification Service?**

Entrust Identification Service **comes from the Entrust Security Transaction Platform**. This platform allows companies to control the identities that are trusted to perform transactions for Web services transactions.

## **12) What UDDI means?**

UDDI stands for **Universal, Description, Discovery, and Integration**. It is the discovery layer in the web services protocol stack.

## **13) Define Entrust Entitlements Service?**

This service **verifies entities that attempt to access a web service**. For Example, the authentication service, the Entitlements Service ensures security in business operations.

## **14) Define Entrust Privacy Service?**

As its name implies, **it deals with security and confidentiality**. This service encrypts data to ensure that only concerned parties can access the data.

**15) What do you mean by PKI?**

It means **Public-Key Infrastructure**.

**16) What tools are used to test a web service?**

I have used **SoapUI** for SOAP WS and Firefox poster plugin for RESTful Services.

**17) Differentiate between a SOA and a Web service?**

SOA is a **design and architecture to implement other services**. SOA can be easily implemented using various protocols such as **HTTP, HTTPS, JMS, SMTP, RMI, IIOP, RPC etc**. While Web service, **itself is an implemented technology**. In fact one can implement SOA using the web service.

**18) Discuss various approaches to develop SOAP based web service?**

We can develop SOAP based web service with two different types of approaches such as **contract-first** and **contract-last**. In the first approach, **the contract is defined first and then the classes are derived from the contract** while in the later one, **the classes are defined first and then the contract is derived from these classes**.

**19) If you have to choose one approach, then what will be your choice?**

In my point of view, **the first approach that is the contract-first approach is more feasible** as compared to the second one but still it depends on other factors too.

**20) Is there any special application required to access web service?**

No, you don't need to install any special application to access web service. You can access web service from any application **that supports XML based object request and response**.

**21) Can you name few free and commercial implementations for web services?**

The implementations I know are **Apache SOAP, JAX-WS Reference Implementation, JAX-RS Reference Implementation, Metro, Apache CXF, MS.NET and Java 6**.

**22) Name browser that allows access to web service?**

**JavaScript XMLHttpRequest object** is required to access web service via browsers. The browsers that support this object are **Internet Explorer, Safari and Mozilla-based browsers like FireFox**.

**23) What is REST?**

REST stands for **Representational State Transfer**. REST itself is not a standard, while it **uses various standards such as HTTP, URL, XML/HTML/GIF/JPEG (Resource Representations)** and **text/xml, text/html, image/gif, image/jpeg, etc (MIME Types)**.

**24) How one can provide API to users?**

To provide an API to the users, one can easily do this with an “open table”. All you need to do is to write open table which is basically an XML schema that point to a web service.

**25) Name the various communication channels in web service?**

Web service is integrated with three protocols such as HTTP/POST, HTTP/GET, and SOAP. It provides three different communication channels to clients. Client can choose any communication method as per requirements.

**26) How can you document web service?**

Web services are contemplated as self-documenting because they provide entire information regarding the available methods and parameters used for XML based standard, known as WSDL. One can also provide more information to explain web services via their own WebService and WebMethod attributes.

**27) What are the situations, when we need ASP.NET web services?**

ASP.NET web services are used when one need to implement three tier architecture in a web service. It allows handy ways to use middle tier components through internet. The main advantage of .NET Web services is that they are capable enough to communicate across firewalls because they use SOAP as transport protocol.

**28) What are distributed technologies?**

The increasing ratio of distributed applications has raised demand for distributed technologies. It allows segmenting of application units and transferring them to different computers on different networks.

**29) Differentiate between web services, CORBA and DCOM?**

Web services transfer/receive messages to/from application respectively, via HTTP protocol. It uses XML to encode data.

CORBA and DCOM transfer/receive messages to/from application respectively, via non-standard protocols such as IIOP and RPC.

**30) Can you tell few benefits of web services?**

The biggest advantage of web service is that is supported by wide variety of platforms. Moreover, in near future, web services may spread its boundary and enhance new methods that will provide ease to clients. The enhancement will not affect the clients, even if they offer old methods and parameters.

**31) Can you name some standards used in web services?**

The standards used in web services are WSDL (used to create interface definition), SOAP (used to structure data), HTTP (communication channels), DISCO (used to create discovery documents) and UDDI (used to create business registries).

**32) Explain in brief, what DISCO is?**

DISCO means discovery. It groups the list of interrelated web services. The organization that provides web services, issues a DISCO file on its server and that file contains the links of all the provided web services. This standard is good when client knows the company already. Also it can be used within a local network as well.

**33) Explain in brief, what UDDI is?**

UDDI (Universal Description, Discovery, and Integration) provides consolidated directory for web services on the internet. Clients use UDDI to find web services as per their business needs. It basically hosts the web services from various companies. In order to share web services, you need to publish it in UDDI.

**34) Explain the .NET web services supported data types?**

.Net web services uses XML-based standards to transfer/receive information. Thus, .NET web services can only work with data types known by XML schema standard. Like FileStream, Eventlog etc. are not recognized by the XML schema standards and hence, not supported in web services.

**35) How a .NET web service is tested?**

ASP.NET uses a test page routinely, when one calls for the URL of .asmx file in any browser. This page shows complete information regarding web services.

**36) How a .NET web service is consumed?**

Since we know that web services are constructed on XML standards. Therefore, clients need to have complete understanding of XML-based messages to interchange messages. Clients can communicate with web services through .NET framework that offers proxy mechanisms. These proxy mechanisms have detailed information regarding data sharing within web services that can be easily used by the clients.

**37) Can you name the two Microsoft solutions for distributed applications?**

The two Microsoft solutions for distributed applications are .NET Web Services and .NET Remoting.

**38) Differentiate between .NET Web Services and .NET Remoting?**

As far as protocol is concerned, .NET Web Service uses HTTP, while, .NET Remoting uses any protocol i.e. TCP/HTTP/SMTP. When it comes to performance, .NET Remoting is comparatively, faster than .NET Web Service. Also, as .NET Web Services are hosted via IIS, therefore, it is far more reliable than the .NET Remoting.

**39) Name the components to be published while deploying a Web Service?**

The components that need to be published during a web service deployment are Web Application Directory, Webservice.asmx File, Webservice.Disco File, Web.Config File and Bin Directory.

#### **40) What are the steps performed by the client to access a web service?**

First of all a web reference to the web service is created by the client in his application. Then a proxy class is generated. After that an object of the proxy class is created and at last, the web service is accessed via that proxy object.

#### **41) How web services are implemented in .NET?**

To implement web services in .NET, HTTP handlers are used that interrupt requests to .asmx files.

#### **42) Explain few disadvantages of Response Caching?**

Response Caching is useless or incompetent when method accepts extensive amount of values because caching means to store lot of information. Also, if the method depends on external source of information, and that are not provided within the parameters then such methods are bypassed.

#### **43) What is the alternate solution to Response Caching?**

One can use Data Caching (System.Web.Caching.Cach) instead of Response Caching.

#### **44) Brief few drawbacks of using GET and POST methods to communicate with the web service?**

These methods are less secure and inhibit users to pass structures and objects as arguments. Also, it doesn't allow users to pass ByRef arguments.

#### **45) How can one access a class as a web service?**

To access a class as a web service, one should inherit the class from the System.Web.Services.WebService class and qualify the class with the WebService attribute.

#### **46) How can one access the web service class method via internet?**

To access web service class method via internet, one should qualify a method with the WebMethod attribute.

#### **47) How a SOAP message is structured?**

A SOAP message is consists of SOAP Envelope, SOAP Headers, and SOAP Body.

#### **48) Can you name different kinds of web services?**

There are two types of web services in total i.e. SOAP based web service and RESTful web service.

This question is already mentioned earlier.

#### **49) What's different in RESTful web services?**

The RESTful web services contains no contract or WSDL file.

**50) Give me few reasons to use RESTful web service?**

The RESTful web services are simple to implement and test. It supports various data formats such as XML, JSON etc.

# TOP 85 JAVASCRIPT

## 1. What is JavaScript?

JavaScript is a client-side scripting language that can be inserted into HTML pages and is understood by web browsers.

## 2. Enumerate the differences between Java and JavaScript?

Java is a complete programming language. In contrast, JavaScript is a coded program that can be introduced to HTML pages. These two languages are not at all inter-dependent and are designed for the different intent. Java is an object – oriented programming (OOPS) or structured programming language like C++ or C whereas JavaScript is a client-side scripting language and it is said to be unstructured programming.

## 3. What are JavaScript types?

Following are the JavaScript types:

- Number
- String
- Boolean
- Function
- Object
- Null
- Undefined

## 4. What is the use of isNaN function?

isNaN function returns true if the argument is not a number otherwise it is false.

## 5. Between JavaScript and an ASP script, which is faster?

JavaScript is faster. JavaScript is a client-side language and thus it does not need the assistance of the web server to execute. On the other hand, ASP is a server-side language and hence is always slower than JavaScript.

```
01 | var band = {
02 | "name": "The Red Hot Chili Peppers",
03 | "members": [
04 | {
05 | "name": "Anthony Kiedis",
06 | "role": "lead vocals"
07 | },
08 | {
09 | "name": "Michael 'Flea' Balzary",
10 | "role": "bass guitar, trumpet, backing vocals"
11 | },
12 | {
13 | "name": "Chad Smith",
14 | "role": "drums,percussion"
15 | },
16 | {
17 | "name": "John Frusciante",
18 | "role": "Lead Guitar"
19 | }
20 |],
21 | "year": "2009"
22 | }
```

## 6. What is negative infinity?

Negative Infinity is a number in JavaScript which can be derived by dividing negative number by zero.

## 7. Is it possible to break JavaScript Code into several lines?

Breaking within a string statement can be done by the use of a backslash, '\', at the end of the first line

Example:

```
1 document.write("This is \a program");
```

And if you change to a new line when not within a string statement, then JavaScript ignores break in line.

Example:

```
1 var x=1, y=2,
2
3 z=
4
5 x+y;
```

The above code is perfectly fine, though not advisable as it hampers debugging.

## 8. Which company developed JavaScript?

Netscape is the software company who developed JavaScript.

## 9. What are undeclared and undefined variables?

Undeclared variables are those that do not exist in a program and are not declared. If the program tries to read the value of an undeclared variable, then a runtime error is encountered.

Undefined variables are those that are declared in the program but have not been given any value. If the program tries to read the value of an undefined variable, an undefined value is returned.

## **10. Write the code for adding new elements dynamically?**

```
1 <html>
2 <head> <title>t1</title>
3 <script type="text/javascript">
4 function addNode() { var newP = document.createElement("p");
5 var textNode = document.createTextNode(" This is a new text node");
6 newP.appendChild(textNode); document.getElementById("firstP").appendChild(newP); }
7 </script> </head>
8 <body> <p id="firstP">firstP</p> </body>
9 </html>
```

## **11. What are global variables? How are these variable declared and what are the problems associated with using them?**

Global variables are those that are available throughout the length of the code, that is, these have no scope. The var keyword is used to declare a local variable or object. If the var keyword is omitted, a global variable is declared.

Example:

```
// Declare a global globalVariable = "Test";
```

The problems that are faced by using global variables are the clash of variable names of local and global scope. Also, it is difficult to debug and test the code that relies on global variables.

## **12. What is a prompt box?**

A prompt box is a box which allows the user to enter input by providing a text box. Label and box will be provided to enter the text or number.

## **13. What is ‘this’ keyword in JavaScript?**

‘This’ keyword is used to point at the current object in the code. For instance: If the code is presently at an object created by the help of the ‘new’ keyword, then ‘this’ keyword will point to the object being created.

## **14. Explain the working of timers in JavaScript? Also elucidate the drawbacks of using the timer, if any?**

Timers are used to execute a piece of code at a set time or also to repeat the code in a given interval of time.

This is done by using the functions **setTimeout**, **setInterval** and **clearInterval**.

The **setTimeout(function, delay)** function is used to start a timer that calls a particular function after the mentioned delay. The **setInterval(function, delay)** function is used to repeatedly execute the given function in the mentioned delay and only halts when cancelled. The **clearInterval(id)** function instructs the timer to stop.

Timers are operated within a single thread, and thus events might queue up, waiting to be executed.

## **15. Which symbol is used for comments in Javascript?**

// for Single line comments and

/\* Multi

Line

Comment

\*/

## **16. What is the difference between ViewState and SessionState?**

‘ViewState’ is specific to a page in a session.

‘SessionState’ is specific to user specific data that can be accessed across all pages in the web application.

## **17. What is === operator?**

== is called as strict equality operator which returns true when the two operands are having the same value without any type conversion.

## **18. Explain how can you submit a form using JavaScript?**

To submit a form using JavaScript use document.form[0].submit();

document.form[0].submit();

## **19. Does JavaScript support automatic type conversion?**

Yes JavaScript does support automatic type conversion, it is the common way of type conversion used by JavaScript developers

## **20. How can the style/class of an element be changed?**

It can be done in the following way:

```
1 document.getElementById("myText").style.fontSize = "20?";
```

or

```
1 document.getElementById("myText").className = "anyclass";
```

## **21. Explain how to read and write a file using JavaScript?**

There are two ways to read and write a file using JavaScript

- Using JavaScript extensions                    \* Using a web page and Active X objects

## **22. What are all the looping structures in JavaScript?**

Following are looping structures in Javascript:

- For
- While
- do-while loops

## **23. What is called Variable typing in Javascript?**

Variable typing is used to assign a number to a variable and the same variable can be assigned to a string.

Example

```
1 i = 10;
2
3 i = "string";
```

This is called variable typing.

## **24. How can you convert the string of any base to integer in JavaScript?**

The parseInt() function is used to convert numbers between different bases. parseInt() takes the string to be converted as its first parameter, and the second parameter is the base of the given string.

In order to convert 4F (of base 16) to integer, the code used will be –

```
1 parseInt ("4F", 16);
```

## **25. Explain the difference between “==” and “===”?**

“==” checks only for equality in value whereas “===” is a stricter equality test and returns false if either the value or the type of the two variables are different.

## **26. What would be the result of 3+2+”7”?**

Since 3 and 2 are integers, they will be added numerically. And since 7 is a string, its concatenation will be done. So the result would be 57.

## **27. Explain how to detect the operating system on the client machine?**

In order to detect the operating system on the client machine, the navigator.appVersion string (property) should be used.

## **28. What do mean by NULL in Javascript?**

The NULL value is used to represent no value or no object. It implies no object or null string, no valid boolean value, no number and no array object.

## **29. What is the function of delete operator?**

The functionality of delete operator is used to **delete all variables and objects in a program but it cannot delete variables declared with VAR keyword.**

## **30. What is an undefined value in JavaScript?**

Undefined value means the

- Variable used in the code doesn't exist
- Variable is not assigned to any value
- Property doesn't exist

## **31. What are all the types of Pop up boxes available in JavaScript?**

- **Alert**
- **Confirm and**
- **Prompt**

## **32. What is the use of Void(0)?**

Void(0) is used to prevent the page from refreshing and parameter "zero" is passed while calling.

Void(0) is **used to call another method without refreshing the page.**

## **33. How can a page be forced to load another page in JavaScript?**

The following code has to be inserted to achieve the desired effect:

```
1 <script language="JavaScript" type="text/javascript" >
2
3 <!-- location.href="http://newhost/newpath/newfile.html"; //--></script>
```

## **34. What is the data type of variables in JavaScript?**

All variables in the JavaScript are **object data types.**

## **35. What is the difference between an alert box and a confirmation box?**

An alert box **displays only one button** which is the OK button.

But a Confirmation box **displays two buttons** namely OK and cancel.

## **36. What are escape characters?**

Escape characters (Backslash) is used when working with special characters like single quotes, double quotes, apostrophes and ampersands. Place backslash before the characters to make it display.

Example:

I m a good boyI m a "good" boy

```
1 document.write "I m a "good" boy"
2 document.write "I m a \"good\" boy"
3
```

## **37. What are JavaScript Cookies?**

Cookies are the small test files stored in a computer and it gets created when the user visits the websites to store information that they need. Example could be User Name details and shopping cart information from the previous visits.

## **38. Explain what is pop() method in JavaScript?**

The pop() method is similar as the shift() method but the difference is that the Shift method works at the end of the array. Also the pop() method take the last element off of the given array and returns it. The array on which is called is then altered.

## **39. Whether JavaScript has concept level scope?**

No. JavaScript does not have concept level scope. The variable declared inside the function has scope inside the function.

## **40. Mention what is the disadvantage of using innerHTML in JavaScript?**

If you use innerHTML in JavaScript the disadvantage is

- Content is replaced everywhere
- We cannot use like “appending to innerHTML”
- Even if you use +=like “innerHTML = innerHTML + ‘html’” still the old content is replaced by html
- The entire innerHTML content is re-parsed and build into elements, therefore its much slower
- The innerHTML does not provide validation and therefore we can potentially insert valid and broken HTML in the document and break it

## **41. What is break and continue statements?**

Break statement exits from the current loop.

Continue statement continues with next statement of the loop.

## **42. What are the two basic groups of datatypes in JavaScript?**

They are as –

- **Primitive**
- **Reference types.**

Primitive types are **number** and **Boolean** data types. Reference types are **more complex types like strings and dates.**

## **43. How generic objects can be created?**

Generic objects can be created as:

```
1 var I = new object();
```

## **44. What is the use of type of operator?**

‘Typeof’ is an operator which is **used to return a string description of the type of a variable.**

## **45. Which keywords are used to handle exceptions?**

**Try... Catch—finally** is used to handle exceptions in the JavaScript

```
1 Try{
2
3 Code
4
5 }
6
7 Catch(exp){
8
9 Code to throw an exception
10
11 }
12
13 Finally{
14
15 Code runs either it finishes successfully or after catch
16
17 }
```

## **46. Which keyword is used to print the text in the screen?**

**document.write(“Welcome”)** is used to print the text – Welcome in the screen.

## **47. What is the use of blur function?**

Blur function is **used to remove the focus from the specified object.**

## **48. What is variable typing?**

Variable typing is used to assign a number to a variable and then assign string to the same variable. Example is as follows:

```
1 i= 8;
2
3 i="john";
```

## **49. How to find operating system in the client machine using JavaScript?**

The ‘**Navigator.appversion**’ is used to find the name of the operating system in the client machine.

## **50. What are the different types of errors in JavaScript?**

There are three types of errors:

- **Load time errors:** Errors which come up **when loading a web page** like improper syntax errors are known as Load time errors and it generates the errors dynamically.
- **Run time errors:** Errors that come due to **misuse of the command inside the HTML language**.
- **Logical Errors:** These are the errors that occur due to the **bad logic performed on a function** which is having different operation.

## **51. What is the use of Push method in JavaScript?**

The push method is **used to add or append one or more elements to the end of an Array**. Using this method, we can append multiple elements by passing multiple arguments

## **52. What is unshift method in JavaScript?**

Unshift method **is like push method which works at the beginning of the array**. This method is used to prepend one or more elements to the beginning of the array.

## **53. What is the difference between JavaScript and Jscript?**

Both are almost similar. JavaScript is **developed by Netscape** and Jscript was **developed by Microsoft**.

## **54. How are object properties assigned?**

Properties are assigned to objects in the following way –

```
1 obj["class"] = 12;
```

or

```
1 obj.class = 12;
```

## **55. What is the ‘Strict’ mode in JavaScript and how can it be enabled?**

Strict Mode adds certain compulsions to JavaScript. Under the strict mode, JavaScript shows errors for a piece of codes, which did not show an error before, but might be problematic and potentially unsafe. Strict mode also solves some mistakes that hamper the JavaScript engines to work efficiently.

Strict mode can be enabled by adding the string literal “use strict” above the file. This can be illustrated by the given example:

```
1 function myfunction()
2
3 {
4
5 "use strict";
6
7 var v = "This is a strict mode function";
8
9 }
```

## **56. What is the way to get the status of a CheckBox?**

The status can be acquired as follows –

```
alert(document.getElementById('checkbox1').checked);
```

If the CheckBox will be checked, this alert will return TRUE.

## **57. How can the OS of the client machine be detected?**

The navigator.appVersion string can be used to detect the operating system on the client machine.

## **58. Explain window.onload and onDocumentReady?**

The onload function is not run until all the information on the page is loaded. This leads to a substantial delay before any code is executed.

onDocumentReady loads the code just after the DOM is loaded. This allows early manipulation of the code.

## **59. How will you explain closures in JavaScript? When are they used?**

Closure is a locally declared variable related to a function which stays in memory when the function has returned.

For example:

```
1 function greet(message) {
2
3 console.log(message);
4 }
```

```

5 }
6
7 function greeter(name, age) {
8
9 return name + " says howdy!! He is " + age + " years old";
10
11 }
12
13 // Generate the message
14
15 var message = greeter("James", 23);
16
17 // Pass it explicitly to greet
18
19 greet(message);
20
21 This function can be better represented by using closures
22
23 function greeter(name, age) {
24
25 var message = name + " says howdy!! He is " + age + " years old";
26
27 return function greet() {
28
29 console.log(message);
30
31 };
32
33 }
34
35 // Generate the closure
36
37 var JamesGreeter = greeter("James", 23);
38
39 // Use the closure
40
41 JamesGreeter();

```

## **60. How can a value be appended to an array?**

A value can be appended to an array in the given manner –

**arr[arr.length] = value;**

## **61. Explain the for-in loop?**

The for-in loop is **used to loop through the properties of an object.**

The syntax for the for-in loop is –

```

1 for (variable name in object){
2

```

```
3 statement or block to execute
```

```
4
```

```
5 }
```

In each repetition, one property from the object is associated to the variable name, and the loop is continued till all the properties of the object are depleted.

## 62. Describe the properties of an anonymous function in JavaScript?

A **function that is declared without any named identifier** is known as an anonymous function. In general, an anonymous function is inaccessible after its declaration.

Anonymous function declaration –

```
1 var anon = function() {
2
3 alert('I am anonymous');
4
5 };
6
7 anon();
```

## 63. What is the difference between .call() and .apply()

The function .call() and .apply() are very similar in their usage except a little difference. .call() is **used when the number of the function's arguments are known to the programmer, as they have to be mentioned as arguments in the call statement**. On the other hand, .apply() is **used when the number is not known. The function .apply() expects the argument to be an array**.

The basic difference between .call() and .apply() is in the way arguments are passed to the function. Their usage can be illustrated by the given example.

```
1 var someObject = {
2
3 myProperty : 'Foo',
4
5 myMethod : function(prefix, postfix) {
6
7 alert(prefix + this.myProperty + postfix);
8
9 }
10
11 };
12
13 someObject.myMethod('<', '>>'); // alerts '<Foo>'
14
15 var someOtherObject = {
16
17 myProperty : 'Bar'
```

```
18
19 };
20
21 someObject.myMethod.call(someOtherObject, '<', '>'); // alerts '<Bar>'
22
23 someObject.myMethod.apply(someOtherObject, ['<', '>']); // alerts '<Bar>'
```

#### 64. Define event bubbling?

##### Document Object Model

JavaScript allows **DOM elements to be nested inside each other**. In such a case, **if the handler of the child is clicked, the handler of parent will also work as if it were clicked too.**

#### 65. Is JavaScript case sensitive? Give an example?

Yes, JavaScript is **case sensitive**. For example, a function `parseInt` is not same as the function `Parseint`.

#### 66. What boolean operators can be used in JavaScript?

The '**And**' Operator (`&&`), '**Or**' Operator (`||`) and the '**Not**' Operator (`!`) can be used in JavaScript.

\*Operators are without the parenthesis.

#### 67. How can a particular frame be targeted, from a hyperlink, in JavaScript?

This can be done by including the name of the required frame in the hyperlink using the 'target' attribute.

```
1 >New Page
```

#### 68. What is the role of break and continue statements?

Break statement is used to come out of the current loop while the continue statement continues the current loop with a new recurrence.

#### 69. Write the point of difference between web-garden and a web-farm?

Both web-garden and web-farm **are web hosting systems**. The only difference is that web-garden is a setup that includes **many processors in a single server** while web-farm is a **larger setup that uses more than one server**.

#### 70. How are object properties assigned?

Assigning properties to objects is done in the same way as a value is assigned to a variable. For example, a form object's action value is assigned as 'submit' in the following manner – `Document.form.action="submit"`

#### 71. What is the method for reading and writing a file in JavaScript?

This can be done by Using JavaScript extensions (runs from JavaScript Editor), example for opening of a file –

```
1 fh = fopen(getScriptPath(), 0);
```

## **72. How are DOM utilized in JavaScript?**

DOM stands for Document Object Model and is responsible for how various objects in a document interact with each other. DOM is required for developing web pages, which includes objects like paragraph, links, etc. These objects can be operated to include actions like add or delete. DOM is also required to add extra capabilities to a web page. On top of that, the use of API gives an advantage over other existing models.

## **73. How are event handlers utilized in JavaScript?**

Events are the actions that result from activities, such as clicking a link or filling a form, by the user. An event handler is required to manage proper execution of all these events. Event handlers are an extra attribute of the object. This attribute includes event's name and the action taken if the event takes place.

## **74. Explain the role of deferred scripts in JavaScript?**

By default, the parsing of the HTML code, during page loading, is paused until the script has not stopped executing. It means, if the server is slow or the script is particularly heavy, then the webpage is displayed with a delay. While using Deferred, scripts delays execution of the script till the time HTML parser is running. This reduces the loading time of web pages and they get displayed faster.

## **75. What are the various functional components in JavaScript?**

The different functional components in JavaScript are-

**First-class functions:** Functions in JavaScript are utilized as first class objects. This usually means that these functions can be passed as arguments to other functions, returned as values from other functions, assigned to variables or can also be stored in data structures.

**Nested functions:** The functions, which are defined inside other functions, are called Nested functions. They are called ‘everytime’ the main function is invoked.

## **76. Write about the errors shown in JavaScript?**

JavaScript gives a message if it encounters an error. The recognized errors are –

- Load-time errors: The errors shown at the time of the page loading are counted under Load-time errors. These errors are encountered by the use of improper syntax, and thus are detected while the page is getting loaded.
- Run-time errors: This is the error that comes up while the program is running. It is caused by illegal operations, for example, division of a number by zero, or trying to access a non-existent area of the memory.
- Logic errors: It is caused by the use of syntactically correct code, which does not fulfill the required task. For example, an infinite loop.

## 77. What are Screen objects?

Screen objects are used to read the information from the client's screen. The properties of screen objects are –

- **AvalHeight:** Gives the height of client's screen
- **AvailWidth:** Gives the width of client's screen.
- **ColorDepth:** Gives the bit depth of images on the client's screen
- **Height:** Gives the total height of the client's screen, including the taskbar
- **Width:** Gives the total width of the client's screen, including the taskbar

## 78. Explain the unshift() method ?

This method is functional at the starting of the array, unlike the push(). It adds the desired number of elements to the top of an array. For example –

```
1 var name = ["john"];
2
3 name.unshift("charlie");
4
5 name.unshift("joseph", "Jane");
6
7 console.log(name);
```

The output is shown below:

```
1 [" Jane ", " joseph ", " charlie ", " john "]
```

## 79. Define unescape() and escape() functions?

The escape () function is responsible for coding a string so as to make the transfer of the information from one computer to the other, across a network.

For Example:

```
1 <script>
2
3 document.write(escape("Hello? How are you!"));
4
5 </script>
```

Output: Hello%3F%20How%20are%20you%21

The unescape() function is very important as it decodes the coded string.

It works in the following way. For example:

```
1 <script>
2
3 document.write(unescape("Hello%3F%20How%20are%20you%21"));
4
```

```
5 </script>
```

Output: Hello? How are you!

#### 80. What are the decodeURI() and encodeURI()?

EncodeURI() is used to convert URL into their hex coding. And DecodeURI() is used to convert the encoded URL back to normal.

```
1 <script>
2
3 var uri="my test.asp?name=st le&car=saab";
4
5 document.write(encodeURI(uri)+"
");
6
7 document.write(decodeURI(uri));
8
9 </script>
```

Output –

```
my%20test.asp?name=st%C3%A5le&car=saab
my test.asp?name=st le&car=saab
```

#### 81. Why it is not advised to use innerHTML in JavaScript?

innerHTML content is refreshed every time and thus is slower. There is no scope for validation in innerHTML and, therefore, it is easier to insert rogue code in the document and, thus, make the web page unstable.

#### 82. What does the following statement declares?

```
1 var myArray = [[[]]];
```

It declares a three dimensional array.

#### 83. How are JavaScript and ECMA Script related?

ECMAScript is nothing but another name for JavaScript. Precisely, ECMAScript is the formal name of JavaScript, when XML elements have to be accessed.

#### 84. What is namespacing in JavaScript and how is it used?

Namespacing is used for grouping the desired functions, variables etc. under a unique name. It is a name that has been attached to the desired functions, objects and properties. This improves modularity in the coding and enables code reuse.

#### 85. How can JavaScript codes be hidden from old browsers that don't support JavaScript?

For hiding JavaScript codes from old browsers:

Add “`<!--`” without the quotes in the code just after the `<script>` tag.

Add “`//-->`” without the quotes in the code just before the `<script>` tag.

Old browsers will now treat this JavaScript code as a long HTML comment. While, a browser that supports JavaScript, will take the “`<!--`” and “`//-->`” as one-line comments.

# TOP 100 C Programming

## 1) How do you construct an increment statement or decrement statement in C?

There are actually two ways you can do this. One is to use the increment operator `++` and decrement operator `-`. For example, the statement “`x++`” means to increment the value of `x` by 1. Likewise, the statement “`x -`” means to decrement the value of `x` by 1. Another way of writing increment statements is to use the conventional `+` plus sign or `-` minus sign. In the case of “`x++`”, another way to write it is “`x = x + 1`”.

## 2) What is the difference between Call by Value and Call by Reference?

When using Call by Value, you are sending the value of a variable as parameter to a function, whereas Call by Reference sends the address of the variable. Also, under Call by Value, the value in the parameter is not affected by whatever operation that takes place, while in the case of Call by Reference, values can be affected by the process within the function.

## 3) Some coders debug their programs by placing comment symbols on some codes instead of deleting it. How does this aid in debugging?

Placing comment symbols `/* */` around a code, also referred to as “commenting out”, is a way of isolating some codes that you think maybe causing errors in the program, without deleting the code. The idea is that if the code is in fact correct, you simply remove the comment symbols and continue on. It also saves you time and effort on having to retype the codes if you have deleted it in the first place.

## 4) What is the equivalent code of the following statement in WHILE LOOP format?

- 1 `for (a=1; a<=100; a++)`
- 2
- 3 `printf ("d\n", a * a);`

Answer:

```
1 a=1;
2
3 while (a<=100) {
4
5 printf ("d\n", a * a);
6
7 a++;
8
9 }
```

## **5) What is a stack?**

A stack is one form of a data structure. Data is stored in stacks using the FILO (First In Last Out) approach. At any particular instance, only the top of the stack is accessible, which means that in order to retrieve data that is stored inside the stack, those on the upper part should be extracted first. Storing data in a stack is also referred to as a PUSH, while data retrieval is referred to as a POP.

## **6) What is a sequential access file?**

When writing programs that will store and retrieve data in a file, it is possible to designate that file into different forms. A sequential access file is such that data are saved in sequential order: one data is placed into the file after another. To access a particular data within the sequential access file, data has to be read one data at a time, until the right one is reached.

## **7) What is variable initialization and why is it important?**

This refers to the process wherein a variable is assigned an initial value before it is used in the program. Without initialization, a variable would have an unknown value, which can lead to unpredictable outputs when used in computations or other operations.

## **8 What is spaghetti programming?**



Spaghetti programming refers to codes that tend to get tangled and overlapped throughout the program. This unstructured approach to coding is usually attributed to lack of experience on the part of the programmer.

Spaghetti programming makes a program complex and analyzing the codes difficult, and so must be avoided as much as possible.

## **9) Differentiate Source Codes from Object Codes**

Source codes are codes that were written by the programmer. It is made up of the commands and other English-like keywords that are supposed to instruct the computer what to do. However, computers would not be able to understand source codes. Therefore, source codes are compiled using a compiler. The resulting outputs are

object codes, which are in a format that can be understood by the computer processor. In C programming, source codes are saved with the file extension .C, while object codes are saved with the file extension .OBJ

### **10) In C programming, how do you insert quote characters (' and ") into the output screen?**

This is a common problem for beginners because quotes are normally part of a printf statement. To insert the quote character as part of the output, use the format specifiers \' (for single quote), and \" (for double quote).

### **11) What is the use of a '\0' character?**

It is referred to as a terminating null character, and is used primarily to show the end of a string value.

### **12) What is the difference between the = symbol and == symbol?**

The = symbol is often used in mathematical operations. It is used to assign a value to a given variable. On the other hand, the == symbol, also known as “equal to” or “equivalent to”, is a relational operator that is used to compare two values.

### **13) What is the modulus operator?**

The modulus operator outputs the remainder of a division. It makes use of the percentage (%) symbol. For example: 10 % 3 = 1, meaning when you divide 10 by 3, the remainder is 1.

### **14) What is a nested loop?**

A nested loop is a loop that runs within another loop. Put it in another sense, you have an inner loop that is inside an outer loop. In this scenario, the inner loop is performed a number of times as specified by the outer loop. For each turn on the outer loop, the inner loop is first performed.

### **15) Which of the following operators is incorrect and why? (>=, <=, <>, ==)**

<> is incorrect. While this operator is correctly interpreted as “not equal to” in writing conditional statements, it is not the proper operator to be used in C programming. Instead, the operator != must be used to indicate “not equal to” condition.

### **16) Compare and contrast compilers from interpreters.**

Compilers and interpreters often deal with how program codes are executed. Interpreters execute program codes one line at a time, while compilers take the program as a whole and convert it into object code, before executing it. The key difference here is that in the case of interpreters, a program may encounter syntax errors in the middle of execution, and will stop from there. On the other hand, compilers check the syntax of the entire program and will only proceed to execution when no syntax errors are found.

**17) How do you declare a variable that will hold string values?**

The char keyword can only hold 1 character value at a time. By creating an array of characters, you can store string values in it. Example: “char MyName[50]; ” declares a string variable named MyName that can hold a maximum of 50 characters.

**18) Can the curly brackets { } be used to enclose a single line of code?**

While curly brackets are mainly used to group several lines of codes, it will still work without error if you used it for a single line. Some programmers prefer this method as a way of organizing codes to make it look clearer, especially in conditional statements.

**19) What are header files and what are its uses in C programming?**

Header files are also known as library files. They contain two essential things: the definitions and prototypes of functions being used in a program. Simply put, commands that you use in C programming are actually functions that are defined from within each header files. Each header file contains a set of functions. For example: stdio.h is a header file that contains definition and prototypes of commands like printf and scanf.

**20) What is syntax error?**

Syntax errors are associated with mistakes in the use of a programming language. It maybe a command that was misspelled or a command that must was entered in lowercase mode but was instead entered with an upper case character. A misplaced symbol, or lack of symbol, somewhere within a line of code can also lead to syntax error.

**21) What are variables and it what way is it different from constants?**

Variables and constants may at first look similar in a sense that both are identifiers made up of one character or more characters (letters, numbers and a few allowable symbols). Both will also hold a particular value. Values held by a variable can be altered throughout the program, and can be used in most operations and computations. Constants are given values at one time only, placed at the beginning of a program. This value is not altered in the program. For example, you can assigned a constant named PI and give it a value 3.1415 . You can then use it as PI in the program, instead of having to write 3.1415 each time you need it.

**22) How do you access the values within an array?**

Arrays contain a number of elements, depending on the size you gave it during variable declaration. Each element is assigned a number from 0 to number of elements-1. To assign or retrieve the value of a particular element, refer to the element number. For example: if you have a declaration that says “intscores[5];”, then you have 5 accessible elements, namely: scores[0], scores[1], scores[2], scores[3] and scores[4].

**23) Can I use “int” data type to store the value 32768? Why?**

No. “int” data type is capable of storing values from -32768 to 32767. To store 32768, you can use “long int” instead. You can also use “unsigned int”, assuming you don’t intend to store negative values.

**24) Can two or more operators such as \n and \t be combined in a single line of program code?**

Yes, it's perfectly valid to combine operators, especially if the need arises. For example: you can have a code like ” printf (“Hello\n\n’World\’”) ” to output the text “Hello” on the first line and “World” enclosed in single quotes to appear on the next two lines.

**25) Why is it that not all header files are declared in every C program?**

The choice of declaring a header file at the top of each C program would depend on what commands/functions you will be using in that program. Since each header file contains different function definitions and prototype, you would be using only those header files that would contain the functions you will need. Declaring all header files in every program would only increase the overall file size and load of the program, and is not considered a good programming style.

**26) When is the “void” keyword used in a function?**

When declaring functions, you will decide whether that function would be returning a value or not. If that function will not return a value, such as when the purpose of a function is to display some outputs on the screen, then “void” is to be placed at the leftmost part of the function header. When a return value is expected after the function execution, the data type of the return value is placed instead of “void”.

**27) What are compound statements?**

Compound statements are made up of two or more program statements that are executed together. This usually occurs while handling conditions wherein a series of statements are executed when a TRUE or FALSE is evaluated. Compound statements can also be executed within a loop. Curly brackets { } are placed before and after compound statements.

**28) What is the significance of an algorithm to C programming?**

Before a program can be written, an algorithm has to be created first. An algorithm provides a step by step procedure on how a solution can be derived. It also acts as a blueprint on how a program will start and end, including what process and computations are involved.

**29) What is the advantage of an array over individual variables?**

When storing multiple related data, it is a good idea to use arrays. This is because arrays are named using only 1 word followed by an element number. For example: to store the 10 test results of 1 student, one can use 10 different variable names (grade1, grade2, grade3... grade10). With arrays, only 1 name is used, the rest are accessible through the index name (grade[0], grade[1], grade[2]... grade[9]).

**30) Write a loop statement that will show the following output:**

1  
12  
123  
1234  
12345

Answer:

```
1 for (a=1; a<=5; i++) {
2 for (b=1; b<=a; b++)
3 printf("d",b);
4 printf("\n");
5 }
6
7
8
9
```

**31) What is wrong in this statement? scanf("%d",whatnumber);**

An ampersand & symbol must be placed before the variable name whatnumber. Placing & means whatever integer value is entered by the user is stored at the “address” of the variable name. This is a common mistake for programmers, often leading to logical errors.

**32) How do you generate random numbers in C?**

Random numbers are generated in C using the rand() command. For example: anyNum = rand() will generate any integer number beginning from 0, assuming that anyNum is a variable of type integer.

**33) What could possibly be the problem if a valid function name such as tolower() is being reported by the C compiler as undefined?**

The most probable reason behind this error is that the header file for that function was not indicated at the top of the program. Header files contain the definition and prototype for functions and commands used in a C program. In the case of “tolower()”, the code “#include <ctype.h>” must be present at the beginning of the program.

**34) What are comments and how do you insert it in a C program?**

Comments are a great way to put some remarks or description in a program. It can serve as a reminder on what the program is all about, or a description on why a certain code or function was placed there in the first place. Comments begin with /\* and end by \*/ characters. Comments can be a single line, or can even span several lines. It can be placed anywhere in the program.

### **35) What is debugging?**

Debugging is the process of identifying errors within a program. During program compilation, errors that are found will stop the program from executing completely. At this state, the programmer would look into the possible portions where the error occurred. Debugging ensures the removal of errors, and plays an important role in ensuring that the expected program output is met.

### **36) What does the && operator do in a program code?**

The && is also referred to as AND operator. When using this operator, all conditions specified must be TRUE before the next action can be performed. If you have 10 conditions and all but 1 fails to evaluate as TRUE, the entire condition statement is already evaluated as FALSE.

### **37) In C programming, what command or code can be used to determine if a number is odd or even?**

There is no single command or function in C that can check if a number is odd or even. However, this can be accomplished by dividing that number by 2, then checking the remainder. If the remainder is 0, then that number is even, otherwise, it is odd. You can write it in code as:

```
1 if (num % 2 == 0)
2
3 printf("EVEN");
4
5 else
6
7 printf("ODD");
```

### **38) What does the format %10.2 mean when included in a printf statement?**

This format is used for two things: to set the number of spaces allotted for the output number and to set the number of decimal places. The number before the decimal point is for the allotted space, in this case it would allot 10 spaces for the output number. If the number of space occupied by the output number is less than 10, addition space characters will be inserted before the actual output number. The number after the decimal point sets the number of decimal places, in this case, it's 2 decimal spaces.

### **39) What are logical errors and how does it differ from syntax errors?**

Program that contains logical errors tend to pass the compilation process, but the resulting output may not be the expected one. This happens when a wrong formula was inserted into the code, or a wrong sequence of

commands was performed. Syntax errors, on the other hand, deal with incorrect commands that are misspelled or not recognized by the compiler.

#### **40) What are the different types of control structures in programming?**

There are 3 main control structures in programming: Sequence, Selection and Repetition. Sequential control follows a top to bottom flow in executing a program, such that step 1 is first perform, followed by step 2, all the way until the last step is performed. Selection deals with conditional statements, which mean codes are executed depending on the evaluation of conditions as being TRUE or FALSE. This also means that not all codes may be executed, and there are alternative flows within. Repetitions are also known as loop structures, and will repeat one or two program statements set by a counter.

#### **41) What is || operator and how does it function in a program?**

The || is also known as the OR operator in C programming. When using || to evaluate logical conditions, any condition that evaluates to TRUE will render the entire condition statement as TRUE.

#### **42) Can the “if” function be used in comparing strings?**

No. “if” command can only be used to compare numerical values and single character values. For comparing string values, there is another function called strcmp that deals specifically with strings.

#### **43) What are preprocessor directives?**

Preprocessor directives are placed at the beginning of every C program. This is where library files are specified, which would depend on what functions are to be used in the program. Another use of preprocessor directives is the declaration of constants. Preprocessor directives begin with the # symbol.

#### **44) What will be the outcome of the following conditional statement if the value of variable s is 10?**

**s >=10 && s < 25 && s!=12**

The outcome will be TRUE. Since the value of s is 10, s >= 10 evaluates to TRUE because s is not greater than 10 but is still equal to 10. s< 25 is also TRUE since 10 is less than 25. Just the same, s!=12, which means s is not equal to 12, evaluates to TRUE. The && is the AND operator, and follows the rule that if all individual conditions are TRUE, the entire statement is TRUE.

#### **45) Describe the order of precedence with regards to operators in C.**

Order of precedence determines which operation must first take place in an operation statement or conditional statement. On the top most level of precedence are the unary operators !, +, - and &. It is followed by the regular mathematical operators (\*, / and modulus % first, followed by + and -). Next in line are the relational operators <, <=, >= and >. This is then followed by the two equality operators == and !=. The logical operators && and || are next evaluated. On the last level is the assignment operator =.

**46) What is wrong with this statement? myName = “Robin”;**

You cannot use the = sign to assign values to a string variable. Instead, use the strcpy function. The correct statement would be: strcpy(myName, “Robin”);

**47) How do you determine the length of a string value that was stored in a variable?**

To get the length of a string value, use the function strlen(). For example, if you have a variable named FullName, you can get the length of the stored string value by using this statement: I = strlen(FullName); the variable I will now have the character length of the string value.

**48) Is it possible to initialize a variable at the time it was declared?**

Yes, you don't have to write a separate assignment statement after the variable declaration, unless you plan to change it later on. For example: char planet[15] = “Earth”; does two things: it declares a string variable named planet, then initializes it with the value “Earth”.

**49) Why is C language being considered a middle level language?**

This is because C language is rich in features that make it behave like a high level language while at the same time can interact with hardware using low level methods. The use of a well structured approach to programming, coupled with English-like words used in functions, makes it act as a high level language. On the other hand, C can directly access memory structures similar to assembly language routines.

**50) What are the different file extensions involved when programming in C?**

Source codes in C are saved with .C file extension. Header files or library files have the .H file extension. Every time a program source code is successfully compiled, it creates an .OBJ object file, and an executable .EXE file.

**51) What are reserved words?**

Reserved words are words that are part of the standard C language library. This means that reserved words have special meaning and therefore cannot be used for purposes other than what it is originally intended for. Examples of reserved words are int, void, and return.

**52) What are linked list?**

A linked list is composed of nodes that are connected with another. In C programming, linked lists are created using pointers. Using linked lists is one efficient way of utilizing memory for storage.

**53) What is FIFO?**

In C programming, there is a data structure known as queue. In this structure, data is stored and accessed using FIFO format, or First-In-First-Out. A queue represents a line wherein the first data that was stored will be the first one that is accessible as well.

**54) What are binary trees?**

Binary trees are actually an extension of the concept of linked lists. A binary tree has two pointers, a left one and a right one. Each side can further branch to form additional nodes, which each node having two pointers as well.

**55) Not all reserved words are written in lowercase. TRUE or FALSE?**

FALSE. All reserved words must be written in lowercase; otherwise the C compiler would interpret this as unidentified and invalid.

**56) What is the difference between the expression “++a” and “a++”?**

In the first expression, the increment would happen first on variable a, and the resulting value will be the one to be used. This is also known as a prefix increment. In the second expression, the current value of variable a would be the one to be used in an operation, before the value of a itself is incremented. This is also known as postfix increment.

**57) What would happen to X in this expression: X += 15; (assuming the value of X is 5)**

X +=15 is a short method of writing X = X + 15, so if the initial value of X is 5, then  $5 + 15 = 20$ .

**58) In C language, the variables NAME, name, and Name are all the same. TRUE or FALSE?**

FALSE. C language is a case sensitive language. Therefore, NAME, name and Name are three uniquely different variables.

**59) What is an endless loop?**

An endless loop can mean two things. One is that it was designed to loop continuously until the condition within the loop is met, after which a break function would cause the program to step out of the loop. Another idea of an endless loop is when an incorrect loop condition was written, causing the loop to run erroneously forever. Endless loops are oftentimes referred to as infinite loops.

**60) What is a program flowchart and how does it help in writing a program?**

A flowchart provides a visual representation of the step by step procedure towards solving a given problem. Flowcharts are made of symbols, with each symbol in the form of different shapes. Each shape may represent a particular entity within the entire program structure, such as a process, a condition, or even an input/output phase.

**61) What is wrong with this program statement? void = 10;**

The word void is a reserved word in C language. You cannot use reserved words as a user-defined variable.

## **62) Is this program statement valid? INT = 10.50;**

Assuming that INT is a variable of type float, this statement is valid. One may think that INT is a reserved word and must not be used for other purposes. However, recall that reserved words are express in lowercase, so the C compiler will not interpret this as a reserved word.

## **63) What are actual arguments?**

When you create and use functions that need to perform an action on some given values, you need to pass these given values to that function. The values that are being passed into the called function are referred to as actual arguments.

## **64) What is a newline escape sequence?**

A newline escape sequence is represented by the \n character. This is used to insert a new line when displaying data in the output screen. More spaces can be added by inserting more \n characters. For example, \n\n would insert two spaces. A newline escape sequence can be placed before the actual output expression or after.

## **65) What is output redirection?**

It is the process of transferring data to an alternative output source other than the display screen. Output redirection allows a program to have its output saved to a file. For example, if you have a program named COMPUTE, typing this on the command line as COMPUTE >DATA can accept input from the user, perform certain computations, then have the output redirected to a file named DATA, instead of showing it on the screen.

## **66) What are run-time errors?**

These are errors that occur while the program is being executed. One common instance wherein run-time errors can happen is when you are trying to divide a number by zero. When run-time errors occur, program execution will pause, showing which program line caused the error.

## **67) What is the difference between functions abs() and fabs()?**

These 2 functions basically perform the same action, which is to get the absolute value of the given value. Abs() is used for integer values, while fabs() is used for floating type numbers. Also, the prototype for abs() is under <stdlib.h>, while fabs() is under <math.h>.

## **68) What are formal parameters?**

In using functions in a C program, formal parameters contain the values that were passed by the calling function. The values are substituted in these formal parameters and used in whatever operations as indicated within the main body of the called function.

## **69) What are control structures?**

Control structures take charge at which instructions are to be performed in a program. This means that program flow may not necessarily move from one statement to the next one, but rather some alternative portions may need to be pass into or bypassed from, depending on the outcome of the conditional statements.

**70) Write a simple code fragment that will check if a number is positive or negative.**

```
1 If (num>=0)
2
3 printf("number is positive");
4
5 else
6
7 printf ("number is negative");
```

**71) When is a “switch” statement preferable over an “if” statement?**

The switch statement is best used when dealing with selections based on a single variable or expression. However, switch statements can only evaluate integer and character data types.

**72) What are global variables and how do you declare them?**

Global variables are variables that can be accessed and manipulated anywhere in the program. To make a variable global, place the variable declaration on the upper portion of the program, just after the preprocessor directives section.

**73) What are enumerated types?**

Enumerated types allow the programmer to use more meaningful words as values to a variable. Each item in the enumerated type variable is actually associated with a numeric code. For example, one can create an enumerated type variable named DAYS whose values are Monday, Tuesday... Sunday.

**74) What does the function toupper() do?**

It is used to convert any letter to its upper case mode. Toupper() function prototype is declared in <ctype.h>. Note that this function will only convert a single character, and not an entire string.

**75) Is it possible to have a function as a parameter in another function?**

Yes, that is allowed in C programming. You just need to include the entire function prototype into the parameter field of the other function where it is to be used.

**76) What are multidimensional arrays?**

Multidimensional arrays are capable of storing data in a two or more dimensional structure. For example, you can use a 2 dimensional array to store the current position of pieces in a chess game, or position of players in a tic-tac-toe program.

**77) Which function in C can be used to append a string to another string?**

The strcat function. It takes two parameters, the source string and the string value to be appended to the source string.

**78) What is the difference between functions getch() and getche()?**

Both functions will accept a character input value from the user. When using getch(), the key that was pressed will not appear on the screen, and is automatically captured and assigned to a variable. When using getche(), the key that was pressed by the user will appear on the screen, while at the same time being assigned to a variable.

**79) Do these two program statements perform the same output? 1) scanf("%c", &letter); 2) letter=getchar()**

Yes, they both do the exact same thing, which is to accept the next key pressed by the user and assign it to variable named letter.

**80) What are structure types in C?**

Structure types are primarily used to store records. A record is made up of related fields. This makes it easier to organize a group of related data.

**81) What does the characters “r” and “w” mean when writing programs that will make use of files?**

“r” means “read” and will open a file as input wherein data is to be retrieved. “w” means “write”, and will open a file for output. Previous data that was stored on that file will be erased.

**82) What is the difference between text files and binary files?**

Text files contain data that can easily be understood by humans. It includes letters, numbers and other characters. On the other hand, binary files contain 1s and 0s that only computers can interpret.

**83) Is it possible to create your own header files?**

Yes, it is possible to create a customized header file. Just include in it the function prototypes that you want to use in your program, and use the #include directive followed by the name of your header file.

**84) What is dynamic data structure?**

Dynamic data structure provides a means for storing data more efficiently into memory. Using dynamic memory allocation, your program will access memory spaces as needed. This is in contrast to static data structure, wherein the programmer has to indicate a fix number of memory space to be used in the program.

**85) What are the different data types in C?**

The basic data types are int, char, and float. Int is used to declare variables that will be storing integer values. Float is used to store real numbers. Char can store individual character values.

**86) What is the general form of a C program?**

A C program begins with the preprocessor directives, in which the programmer would specify which header file and what constants (if any) to be used. This is followed by the main function heading. Within the main function lies the variable declaration and program statement.

**87) What is the advantage of a random access file?**

If the amount of data stored in a file is fairly large, the use of random access will allow you to search through it quicker. If it had been a sequential access file, you would have to go through one record at a time until you reach the target data. A random access file lets you jump directly to the target address where data is located.

**88) In a switch statement, what will happen if a break statement is omitted?**

If a break statement was not placed at the end of a particular case portion? It will move on to the next case portion, possibly causing incorrect output.

**89) Describe how arrays can be passed to a user defined function**

One thing to note is that you cannot pass the entire array to a function. Instead, you pass to it a pointer that will point to the array first element in memory. To do this, you indicate the name of the array without the brackets.

**90) What are pointers?**

Pointers point to specific areas in the memory. Pointers contain the address of a variable, which in turn may contain a value or even an address to another memory.

**91) Can you pass an entire structure to functions?**

Yes, it is possible to pass an entire structure to a function in a call by method style. However, some programmers prefer declaring the structure globally, then pass a variable of that structure type to a function. This method helps maintain consistency and uniformity in terms of argument type.

**92) What is gets() function?**

The gets() function allows a full line data entry from the user. When the user presses the enter key to end the input, the entire line of characters is stored to a string variable. Note that the enter key is not included in the variable, but instead a null terminator \0 is placed after the last character.

**93) The % symbol has a special use in a printf statement. How would you place this character as part of the output on the screen?**

You can do this by using %% in the printf statement. For example, you can write printf("10%%") to have the output appear as 10% on the screen.

**94) How do you search data in a data file using random access method?**

Use the fseek() function to perform random access input/output on a file. After the file was opened by the fopen() function, the fseek would require three parameters to work: a file pointer to the file, the number of bytes to search, and the point of origin in the file.

**95) Are comments included during the compilation stage and placed in the EXE file as well?**

No, comments that were encountered by the compiler are disregarded. Comments are mostly for the guidance of the programmer only and do not have any other significant use in the program functionality.

**96) Is there a built-in function in C that can be used for sorting data?**

Yes, use the qsort() function. It is also possible to create user defined functions for sorting, such as those based on the balloon sort and bubble sort algorithm.

**97) What are the advantages and disadvantages of a heap?**

Storing data on the heap is slower than it would take when using the stack. However, the main advantage of using the heap is its flexibility. That's because memory in this structure can be allocated and remove in any particular order. Slowness in the heap can be compensated if an algorithm was well designed and implemented.

**98) How do you convert strings to numbers in C?**

You can write your own functions to do string to number conversions, or instead use C's built in functions. You can use atof to convert to a floating point value, atoi to convert to an integer value, and atol to convert to a long integer value.

**99) Create a simple code fragment that will swap the values of two variables num1 and num2.**

```
1 int temp;
2
3 temp = num1;
4
5 num1 = num2;
6
7 num2 = temp;
```

**100) what is the use of a semicolon (;) at the end of every program statement?**

It has to do with the parsing process and compilation of the code. A semicolon acts as a delimiter, so that the compiler knows where each statement ends, and can proceed to divide the statement into smaller elements for syntax checking.

**Q: What do you know about Java?**

A: Java is a **high-level programming language** originally developed by **Sun Microsystems** and released in **1995**. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

**Q: What are the supported platforms by Java Programming Language?**

A: Java runs on a variety of platforms, such as **Windows**, Mac OS, and the various versions of **UNIX/Linux**  
**OOAD - Object-Oriented Analysis Design (OOA)** is the procedure of identifying software engineering requirements and developing software specifications in terms of a software system's object model, which comprises of interacting objects.

**Q: List any features of Java?** Some features include **Object Oriented, Platform Independent, Multi-threaded**

**Q: How Java enabled High Performance?**

A: Java uses **Just-In-Time compiler** to enable high performance. Just-In-Time compiler is a **program that turns Java bytecode**, which is a program that contains instructions that must be interpreted into **instructions that can be sent directly to the processor**.

**Q: What is Java Virtual Machine and how it is considered in context of Java's platform independent feature?**

A: When Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This **byte code is distributed over the web** and **interpreted by virtual Machine (JVM)** on whichever platform it is being run.

**Q: List some Java keywords (unlike C, C++ keywords)?**

A: Some Java keywords are **import, super, finally**, etc.

**Q: What do you mean by Object?**

A: Object is a **runtime entity** and its **state is stored in fields** and **behavior is shown via methods**. Methods operate on an object's internal state and serve as the primary mechanism for object-to-object communication.

**Q: Define class?**

A: A class is a **blue print from which individual objects are created**. A class can contain fields and methods to describe the behavior of an object.

**Q: What kind of variables a class can consist of?**

A: A class consists of **Local variable, instance variables and class variables**.

**Q: What is a Local Variable?**

A: **Variables defined inside methods, constructors or blocks** are called local variables.

**Q: What is an Instance Variable?**

A: Instance variables are **variables within a class but outside any method**. These variables are instantiated when the class is loaded.

**Q: What is a Class Variable?**

A: These are variables declared within a class, outside any method, **with the static keyword**.

**Q: What is Singleton class?**

A: Singleton class **control object creation**, limiting the number to one but allowing the flexibility to create more objects if the situation changes.

**Q: What do you mean by Constructor?**

**A:** Constructor gets invoked when a new object is created. Every class has a constructor. If we do not explicitly write a constructor for a class the java compiler builds a default constructor for that class.

**Q: List the three steps for creating an Object for a class?**

**A:** An Object is first declared, then instantiated and then it is initialized.

**Q: What is the default value of byte datatype in Java?**

**A:** Default value of byte datatype is 0.

**Q: What is the default value of float and double datatype in Java?**

**A:** Default value of float and double datatype in different as compared to C/C++. For float its 0.0f and for double its 0.0d

**Q: When a byte datatype is used?**

**A:** This data type is used to save space in large arrays, mainly in place of integers, since a byte is four times smaller than an int.

**Q: What is a static variable?**

**A:** Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.

**Q: What do you mean by Access Modifier?**

**A:** Java provides access modifiers to set access levels for classes, variables, methods and constructors. A member has package or default accessibility when no accessibility modifier is specified.

**Q: What is protected access modifier?**

**A:** Variables, methods and constructors which are declared protected in a superclass can be accessed only by the subclasses in other package or any class within the package of the protected members' class.

**Q: What do you mean by synchronized Non Access Modifier?**

**A:** Java provides these modifiers for providing functionalities other than Access Modifiers, synchronized used to indicate that a method can be accessed by only one thread at a time.

**Q: According to Java Operator precedence, which operator is considered to be with highest precedence?**

**A:** Postfix operators i.e () [] . is at the highest precedence. Lowest is Assignment operators

**Q: Variables used in a switch statement can be used with which datatypes?**

**A:** Variables used in a switch statement can only be a byte, short, int, or char.

**Q: When parseInt() method can be used?**

**A:** This method is used to get the primitive data type of a certain String.

**Q: What is finalize() method?**

**A:** It is possible to define a method that will be called just before an object's final destruction by the garbage collector. This method is called finalize( ), and it can be used to ensure that an object terminates cleanly.

## **Q: What is an Exception?**

**A:** An exception is a problem that arises during the execution of a program. Exceptions are caught by handlers positioned along the thread's method invocation stack.

## **Q: Which are the two subclasses under Exception class?**

**A:** The Exception class has two main subclasses : IOException class and RuntimeException Class.

## **Q: When throws keyword is used?**

**A:** If a method does not handle a checked exception, the method must declare it using the throws keyword. The throws keyword appears at the end of a method's signature.

public void name throws RuntimeException

## **Q: When throw keyword is used?**

**A:** An exception can be thrown, either a newly instantiated one or an exception that you just caught, by using throw keyword.

throw new exception

## **Q: How finally used under Exception Handling?**

**A:** The finally keyword is used to create a block of code that follows a try block. A finally block of code always executes, whether or not an exception has occurred.

## **Q: Define Inheritance?**

**A:** It is the process where one object acquires the properties of another. With the use of inheritance the information is made manageable in a hierarchical order.

## **Q: When super keyword is used?**

**A:** If the method overrides one of its superclass's methods, overridden method can be invoked through the use of the keyword super. It can be also used to refer to a hidden field

## **Q: What is Polymorphism?**

**A:** Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

## **Q: What is Abstraction?**

**A:** It refers to the ability to make a class abstract in OOP. It helps to reduce the complexity and also improves the maintainability of the system.

## **Q: What is Abstract class**

**A:** These classes cannot be instantiated and are either partially implemented or not at all implemented. This class contains one or more abstract methods which are simply method declarations without a body.

## **Q: When Abstract methods are used?**

**A:** If you want a class to contain a particular method but you want the actual implementation of that method to be determined by child classes, you can declare the method in the parent class as abstract.

## **Q: What is Encapsulation?**

**A:** It is the technique of making the fields in a class private and providing access to the fields via public methods. If a field is declared private, it cannot be accessed by anyone outside the class, thereby hiding the fields within the class. Therefore encapsulation is also referred to as data hiding.

**Q: What is the primary benefit of Encapsulation?**

A: The main benefit of encapsulation is the ability to modify our implemented code without breaking the code of others who use our code. With this Encapsulation gives maintainability, flexibility and extensibility to our code.

**Q: What is an Interface?**

A: An interface is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface. **COLLECTIONS:** (lists hold the array stuffs...)

**Q: Define Packages in Java?**

A: A Package can be defined as a grouping of related types (classes, interfaces, enumerations and annotations) providing access protection and name space management.

**Q: What do you mean by multithreaded program?**

A: A multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a thread, and each thread defines a separate path of execution.

**Q: What are the two ways in which Thread can be created?**

A: Thread can be created by: implementing Runnable interface, extending the Thread class.

**Q: What is an applet?**

A: An applet is a Java program that runs in a Web browser. An applet can be a fully functional Java application because it has the entire Java API at its disposal.

**Q: Explain garbage collection in Java?**

A: It uses garbage collection to free the memory. By cleaning those objects that is no longer reference by any of the program.

**Q: Explain the usage of this() with constructors?**

A: It is used with variables or methods and used to call constructor of same class.

**Q: Difference between throw and throws?**

A: It includes:

- Throw is used to trigger an exception whereas throws is used in declaration of exception.
- Without throws, Checked exception cannot be handled whereas checked exception can be propagated with throws.

**Q: Explain the following line used under Java Program:**

**public static void main (String args[ ])**

A: The following shows the explanation individually:

- public: it is the access specifier.
- static: it allows main() to be called without instantiating a particular instance of a class.
- void: it affirms the compiler that no value is returned by main().
- main(): this method is called at the beginning of a Java program.

- String args[ ]: args parameter is an instance array of class String

**Q: Define JRE i.e. Java Runtime Environment?**

A: Java Runtime Environment is an implementation of the Java Virtual Machine which executes Java programs. It provides the minimum requirements for executing a Java application;

**Q: What is JAR file?**

A: JAR files is Java Archive files and it aggregates many files into one. It holds Java classes in a library. JAR files are built on ZIP file format and have .jar file extension.

**Q: What is a WAR file?**

A: This is Web Archive File and used to store XML, java classes, and JavaServer pages. which is used to distribute a collection of JavaServer Pages, Java Servlets, Java classes, XML files, static Web pages etc.

**Q: Define JIT compiler?**

A: It improves the runtime performance of computer programs based on bytecode.

**Q: What is the difference between object oriented programming language and object based programming language?**

A: Object based programming languages follow all the features of OOPs except Inheritance. JavaScript is an example of object based programming languages

**Q: Can a constructor be made final?**

A: No, this is not possible.

**Q: What is static block?**

A: It is used to initialize the static data member, It is executed before main method at the time of class loading.

**Q: Define composition?**

A: Holding the reference of the other class within some other class is known as composition.

**Q: What is function overloading?**

A: If a class has multiple functions by same name but different parameters, it is known as Method Overloading.

**Q: Difference between Overloading and Overriding?**

A: Method overloading increases the readability of the program. Method overriding provides the specific implementation of the method that is already provided by its super class parameter must be different in case of overloading, parameter must be same in case of overriding. wait() sleep()  
wait() method releases the lock. doesn't release the lock of obj method of Object class. is the method of Thread class

**Q: What is the difference between yielding and sleeping?**

A: When a task invokes its yield() method, it returns to the ready state. When a task invokes its sleep() method, it returns to the waiting state. The sleep() method of Thread class is used to sleep a thread for the specified amount of time. If you sleep a thread for the specified time, the thread scheduler picks up another thread and so on.

**Q: Why Vector class is used?**

A: The Vector class provides the capability to implement a growable array of objects. Vector proves to be very useful if you don't know the size of the array in advance, or you just need one that can change sizes over the lifetime of a program.

**Q: What are Wrapper classes?**

A: These are classes that allow primitive types to be accessed as objects. Example: Integer, Character, Double, Boolean etc.

**Q: What is the difference between static and non-static variables?**

A: A static variable is associated with the class as a whole rather than with specific instances of a class. Non-static variables take on unique values with each object instance.

**Q: What is the difference between Swing and AWT components?**

A: AWT components are heavy-weight, whereas Swing components are lightweight. Heavy weight components depend on the local windowing toolkit. For example, java.awt.Button is a heavy weight component, when it is running on the Java platform for Unix platform, it maps to a real Motif button.

**Q: What's the difference between constructors and other methods?**

A: Constructors must have the same name as the class and can not return a value. They are only called once while regular methods could be called many times. **There are two types of thread synchronization mutual exclusive and inter-thread communication.**

**Q: What is synchronization?** Mutual Exclusive - helps keep threads from interfering with one another while sharing data. This can be done by three ways in java:

A: Synchronization is the capability to control the access of multiple threads to shared resources. synchronized keyword in java provides locking which ensures mutual exclusive access of shared resource and prevent data race.

-**Synchronized method.** If you declare any method as synchronized, it is known as synchronized method.

-**Synchronized block.** Suppose you have 50 lines of code in your method, but you want to synchronize only 5 lines, you can use synchronized block.

**Q: What is the difference between a break statement and a continue statement?**

A: A break statement results in the termination of the statement to which it applies (switch, for, do, or while). A continue statement is used to end the current loop iteration and return control to the loop statement.

**Q: What is the difference between an Interface and an Abstract class?**

A: An abstract class can have instance methods that implement a default behavior. An Interface can only declare constants and instance methods, but cannot implement default behavior and all methods are implicitly abstract. An interface has all public members and no implementation.

**Q: What is the difference between error and an exception?**

A: An error is an irrecoverable condition occurring at runtime. Such as OutOfMemory error. Exceptions are conditions that occur because of bad input etc. e.g. FileNotFoundException will be thrown if the specified file does not exist.

**Q: Is it necessary that each try block must be followed by a catch block?**

A: It is not necessary that each try block must be followed by a catch block. It should be followed by either a catch block or a finally block.

**Q: What are synchronized methods and synchronized statements?**

A: Synchronized methods are methods that are used to control access to an object. A synchronized statement can only be executed after a thread has acquired the lock for the object or class referenced in the synchronized statement.

**Q: What is type casting?**

A: Type casting means treating a variable of one type as though it is another type.

**Q: Describe life cycle of thread?**

A: A thread is a execution in a program. The life cycle of a thread include:

- Newborn state
- Runnable state
- Running state
- Blocked state
- Dead state

**Q: What is the difference between the >> and >>> operators?**

A: The >> operator carries the sign bit when shifting right. The >>> zero-fills bits that have been shifted out.

**Q: Does Java allow Default Arguments?**

A: No, Java does not allow Default Arguments.

**Q: Why Generics are used in Java?**

A: Generics provide compile-time type safety that allows programmers to catch invalid types at compile time. Java Generic methods and generic classes enable programmers to specify, with a single method declaration, a set of related methods or, with a single class declaration, a set of related types.

**Q: What is daemon thread? e.g. gc, finalizer etc.**

A: Daemon thread is a low priority thread, which runs intermittently in the back ground doing the garbage collection operation for the java runtime system.

**Q: Which method is used to create the daemon thread?**

A: setDaemon method is used to create a daemon thread.

**Q: What is an enumeration?**

A: An enumeration is an interface containing methods for accessing the underlying data structure from which the enumeration is obtained. It allows sequential access to all the elements stored in the collection.

**Q: Which object oriented Concept is achieved by using overloading and overriding?**

A: Polymorphism

**Q: If a method is declared as protected, where may the method be accessed?**

A: A protected method may only be accessed by classes or interfaces of the same package or by subclasses of the class in which it is declared.

**Q: What is the difference between inner class and nested class?**

A: When a class is defined within a scope of another class, then it becomes inner class. If the access modifier of the inner class is static, then it becomes nested class.

**1. What is the difference between a constructor and a method?**

A constructor is a member function of a class that is used to create objects of that class. It has the same name as the class itself, has no return type, and is invoked using the new operator.

A method is an ordinary member function of a class. It has its own name, a return type (which may be void), and is invoked using the dot operator.

### 9) What is difference between c++ and Java ?

Java	C++
1) Java is platform independent	C++ is platform dependent.
2) There are no pointers in java	There are pointers in C++.
3) There is no operator overloading in java	C ++ has operator overloading.
4) There is garbage collection in java	There is no garbage collection
5) Supports multithreading	Doesn't support multithreading
6) There are no <u>templates</u> in java	There are templates in C++.
7) There are no global variables in java	There are global variables in c++

## What is method

A Java method is a **collection of statements that are grouped together to perform an operation**. When you call the System.out.println method,

## What is a Constant in Java?

A constant in Java is used to map an exact and **unchanging value to a variable name**. Use **final keyword**

**Constructor Chaining:** **Calling a constructor from the another constructor of same class** is known as Constructor chaining. In the below example the class “ChainingDemo” has 4 constructors and we are calling one constructor from another using **this () statement**. For e.g. in order to call a constructor with single string argument we have supplied a string in this () statement like **this(“hello”)**.

## What is meant by pass by reference and pass by value in Java?

Pass by reference means, **passing the address itself** rather than passing the value. Pass by value means **passing a copy of the value**.

What is the difference between Association, Aggregation and Composition?

Association : Association is a relationship between two separate classes which can be of any type say one to one, one to many.  
People uses a browser

Aggregation : one way relationship between classes. Facebook has a user

Composition : restricted form of Aggregation in which two classes are highly dependent on each other For ex: Each user in facebook has a session.

### **Q3. What's the purpose of Static methods and static variables?**

Ans: When there is a requirement to share a method or a variable between multiple objects of a class instead of creating separate copies for each object, we use static keyword to make a method or variable shared for all objects.

### **Q7: What is an infinite Loop? How infinite loop is declared?**

Ans: An infinite loop runs without any condition and runs infinitely. An infinite loop can be broken by defining any breaking logic in the body of the statement blocks.

Infinite loop is declared as follows:



```
1 For (;;) {
2
3 {
4
5 // Statements to execute
6
7 // Add any loop breaking logic
8
9 }
```

### **Q9. What is the difference between double and float variables in Java?**

Ans: In java, float takes 4 bytes in memory while Double takes 8 bytes in memory. Float is single precision floating point decimal number while Double is double precision decimal number.

### **Q11. What is ternary operator? Give an example.**

Ans: Ternary operator , also called conditional operator is used to decide which value to assign to a variable based on a Boolean value evaluation. It's denoted as ?

In the below example, if rank is 1, status is assigned a value of “Done” else “Pending”.



```
1 public class conditionTest {
2 public static void main(String args[]) {
3 String status;
4 int rank;
5 status= (rank == 1) ? "Done": "Pending";
6 }
7 }
```

## **Q12: What are 6 different types of operators in Java?**

Ans: In java, operators can be classified in following six types:

- **Arithmetic** Operators

Used for arithmetic calculations. Example are `+, -, *, /, %, ++, -`

- **Relational** Operators

Used for relational comparison. E.g. `==, !=, >, <, <=, >=`

- **Bitwise** operators

Used for bit by bit operations. E.g. `&, |, ^, ~`

- **Logical** Operators

Used for logical comparisons. E.g. `&&, ||, !`

- **Assignment** Operators

Used for assigning values to variables. E.g. `=, +=, -=, *=, /=`

## **Q14. What's the base class in Java from which all classes are derived?**

Ans: `java.lang.Object`

## **Q15. Can main() method in Java can return any data?**

Ans: In java, main() method `can't return any data` and hence, it's always `declared with a void return type`.

## **Q21. Can we declare the main method of our class as private?**

Ans: In java, main method `must be public static in order to run any application correctly`. If main method is declared as private, developer won't get any compilation error however, it will not get executed and `will give a runtime error`.

## **Q22. How can we pass argument to a function by reference instead of pass by value?**

Ans: In java, we can pass argument to a function `only by value and not by reference`.

## **Q23. How an object is serialized in java?**

Ans: In java, to convert an object into byte stream by serialization, `an interface with the name Serializable is implemented by the class`. All objects of a class implementing serializable interface get serialized and their state is saved in byte stream.

## **Q24. When we should use serialization?**

Ans: Serialization is used when data needs to be transmitted over the network. Using serialization, object's state is saved and converted into byte stream .The byte stream is transferred over the network and the object is re-created at destination.

### **Q28. Can a class have multiple constructors?**

Ans: Yes, a class can have multiple constructors with different parameters. Which constructor gets used for object creation depends on the arguments passed while creating the objects.

### **Q29. Can we override static methods of a class?**

Ans: We cannot override static methods. Static methods belong to a class and not to individual objects and are resolved at the time of compilation (not at runtime).Even if we try to override static method,we will not get an compilation error,nor the impact of overriding when running the code.

### **Q31. Is String a data type in java?**

Ans: String is not a primitive data type in java. When a string is created in java, it's actually an object of Java.Lang.String class that gets created. After creation of this string object, all built-in methods of String class can be used on the string object.

### **Q32. In the below example, how many String Objects are created?**

- 1 String s1="I am Java Expert";
- 2
- 3 String s2="I am C Expert";
- 4
- 5 String s3="I am Java Expert";

Ans: In the above example, two objects of Java.Lang.String class are created. s1 and s3 are references to same object.

### **Q33. Why Strings in Java are called as Immutable?**

Ans: In java, string objects are called immutable as once value has been assigned to a string, it can't be changed and if changed, a new object is created.

In below example, reference str refers to a string object having value "Value one".

- 1 String str="Value One";

When a new value is assigned to it, a new String object gets created and the reference is moved to the new object.

- 1 str="New Value";

### **Q35. What is multi-threading?**

Ans: Multi threading is a programming concept to run multiple tasks in a concurrent manner within a single program. Threads share same process stack and running in parallel. It helps in performance improvement of any program.

### **Q36. Why Runnable Interface is used in Java?**

Ans: Runnable interface is used in java for implementing multi threaded applications. Java.Lang.Runnable interface is implemented by a class to support multi threading.

### **Q38. When a lot of changes are required in data, which one should be a preference to be used? String or StringBuffer?**

Ans: Since StringBuffers are dynamic in nature and we can change the values of StringBuffer objects unlike String which is immutable, it's always a good choice to use StringBuffer when data is being changed too much. If we use String in such a case, for every data change a new String object will be created which will be an extra overhead.

### **Q41. How we can execute any code even before main method?**

Ans: If we want to execute any statements before even creation of objects at load time of class, we can use a static block of code in the class. Any statements inside this static block of code will get executed once at the time of loading the class even before creation of objects in the main method.

### **Q42. Can a class be a super class and a sub-class at the same time? Give example.**

Ans: If there is a hierarchy of inheritance used, a class can be a super class for another class and a sub-class for another one at the same time.

In the example below, continent class is sub-class of world class and it's super class of country class.

```
1 public class world {
2
3
4
5 }
6
7 public class continent extends world {
8
9
10
11 }
12
13 public class country extends continent {
14
15
16
17 }
```

#### **Q44. In multi-threading how can we ensure that a resource isn't used by multiple threads simultaneously?**

Ans: In multi-threading, access to the resources which are shared among multiple threads can be controlled by using the concept of synchronization. Using synchronized keyword, we can ensure that only one thread can use shared resource at a time and others can get control of the resource only once it has become free from the other one using it.

#### **Q45. Can we call the constructor of a class more than once for an object?**

Ans: Constructor is called automatically when we create an object using new keyword. It's called only once for an object at the time of object creation and hence, we can't invoke the constructor again for an object after its creation.

#### **Q46. There are two classes named classA and classB. Both classes are in the same package. Can a private member of classA be accessed by an object of classB?**

Ans: Private members of a class aren't accessible outside the scope of that class and any other class even in the same package can't access them.

#### **Q47. Can we have two methods in a class with the same name?**

Ans: We can define two methods in a class with the same name but with different number/type of parameters. Which method is to get invoked will depend upon the parameters passed.

For example in the class below we have two print methods with same name but different parameters. Depending upon the parameters, appropriate one will be called:

```
1 public class methodExample {
2
3 public void print() {
4
5 System.out.println("Print method without parameters.");
6
7 }
8
9 public void print(String name) {
10
11 System.out.println("Print method with parameter");
12
13 }
14
15 public static void main(String args[]) {
16
17 methodExample obj1=new methodExample();
18
19 obj1.print();
```

```

20
21 obj1.print("xx");
22
23 }
24
25 }
```

#### **Q48. How can we make copy of a java object?**

Ans: We can use the **concept of cloning to create copy of an object**. Using clone, we create copies with the actual state of an object.

**Clone()** is a method of **Cloneable interface** and hence, Cloneable interface needs to be implemented for making object copies.

#### **Q52. How can we restrict inheritance for a class so that no class can be inherited from it?**

Ans: If we want a class not to be extended further by any class, we can use the **keyword Final** with the class name.

In the following example, Stone class is Final and can't be extend

```

1 <pre>
2 public Final Class Stone {
3
4 // Class methods and Variables
5
6 }
```

#### **Q53. What's the access scope of Protected Access specifier?**

Ans When a method or a variable is declared with Protected access specifier, it becomes accessible in the **same class,any other class of the same package as well as a sub-class**.

MODIFIER	CLASS	PACKAGE	SUBCLASS	WORLD
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

*Access Levels*

#### **Q55. In java, how we can disallow serialization of variables?**

Ans: If we want certain variables of a class not to be serialized, we can use the keyword **transient** while declaring them. For example, the variable trans\_var below is a transient variable and can't be serialized:

```
1 public class transientExample {
2
3 private transient trans_var;
4
5 // rest of the code
6
7 }
```

### **Q57. Which types of exceptions are caught at compile time?**

Ans: Checked exceptions can be caught at the time of program compilation. Checked exceptions must be handled by using try catch block in the code in order to successfully compile the code.

### **Q61. A person says that he compiled a java class successfully without even having a main method in it? Is it possible?**

Ans: main method is an entry point of Java class and is required for execution of the program however; a class gets compiled successfully even if it doesn't have a main method. It can't be run though.

### **Q62. Can we call a non-static method from inside a static method?**

Ans: Non-Static methods are owned by objects of a class and have object level scope and in order to call the non-Static methods from a static block (like from a static main method), an object of the class needs to be created first. Then using object reference, these methods can be invoked.

### **Q63. What are the two environment variables that must be set in order to run any Java programs?**

Ans: Java programs can be executed in a machine only once following two environment variables have been properly set:

1. PATH variable
2. CLASSPATH variable

### **Q64. Can variables be used in Java without initialization?**

Ans: In Java, if a variable is used in a code without prior initialization by a valid value, program doesn't compile and gives an error as no default value is assigned to variables in Java.

### **Q65. Can a class in Java be inherited from more than one class?**

Ans: In Java, a class can be derived from only one class and not from multiple classes. Multiple inheritances is not supported by Java.

Why multiple inheritances are not supported in Java?

### **Q67. What will be the output of Round(3.7) and Ceil(3.7)?**

Ans: Round(3.7) returns 3 while Ceil(3.7) returns 4.

The designers of Java considered multiple inheritance to be too complex, and not in line with the goal of keeping Java simple. One of the most common scenario is Diamond problem. Java's alternative to multiple inheritance – interfaces

## **Q69. Can a dead thread be started again?**

Ans: In java, a thread which is in **dead state can't be started again**. There is no way to restart a dead thread.

## **Q72. What's the difference between comparison done by equals method and == operator?**

Ans: In Java, equals() method is used to **compare the contents of two string objects** and returns true if the two have same value while == operator compares the references of two string objects.

In the following example, equals() returns true as **the two string objects have same values**. However == operator returns false as both string objects are referencing to different objects:

```
1 public class equalsTest {
2
3 public static void main(String args[]) {
4
5 String str1="Hello World";
6
7 String str2="Hello World";
8
9 If (str1.equals(str2))
10
11 {// this condition is true
12
13 System.out.println("str1 and str2 are equal in terms of values");
14
15 }
16
17 If (str1==str2) {
18
19 //This condition is not true
20
21 System.out.println("Both strings are referencing same object");
22
23 }
24
25 Else
26
27 {
28
29 // This condition is true
30
31 System.out.println("Both strings are referencing different objects");
32
33 }
34
35 }}
```

## **Q73. Is it possible to define a method in Java class but provide its implementation in the code of another language like C?**

Ans: Yes, we can do this by **use of native methods**. In case of native method based development, we define public static methods in our Java class without its implementation and then implementation is done in another language like C separately.

#### **Q74. How destructors are defined in Java?**

Ans: In Java, there are **no destructors defined** in the class as there is no need to do so. Java has its own garbage collection mechanism which does the job automatically by destroying the objects when no longer referenced.

#### **Q75. Can a variable be local and static at the same time?**

Ans: **No a variable can't be static as well as local at the same time**. Defining a local variable as static gives compilation error.

#### **Q76. Can we have static methods in an Interface?**

Ans: Static methods **can't be overridden in any class** while any **methods in an interface are by default abstract** and are supposed to be implemented in the classes being implementing the interface. So it makes no sense to have static methods in an interface in Java.

#### **Q77. In a class implementing an interface, can we change the value of any variable defined in the interface?**

Ans: **No, we can't change** the value of any variable of an interface in the implementing class as all variables defined in the interface are by default public, static and Final and final variables are like constants which can't be changed later.

#### **Q79. Can we have any other return type than void for main method?**

Ans: **No, Java class main method can have only void return type** for the program to get successfully executed.

Nonetheless , if you absolutely must return a value to at the completion of main method , you can use System.exit(int status)

#### **Q80. I want to re-reach and use an object once it has been garbage collected. How it's possible?**

Ans: Once an object has been destroyed by garbage collector, **it no longer exists on the heap** and it can't be accessed again. There is no way to reference it again.

#### **Q81. In Java thread programming, which method is a must implementation for all threads?**

Ans: **Run()** is a method of **Runnable interface** that must be implemented by all threads.

#### **Q82. I want to control database connections in my program and want that only one thread should be able to make database connection at a time. How can I implement this logic?**

Ans: This can be implemented by use of the concept of synchronization. Database related code can be placed in a method which has **synchronized** keyword so that only one thread can access it at a time.

#### **Q84. I want my class to be developed in such a way that no other class (even derived class) can create its objects. How can I do so?**

Ans: If we declare the constructor of a class as private, it will not be accessible by any other class and hence, no other class will be able to instantiate it and formation of its object will be limited to itself only.

#### **Q85. How objects are stored in Java?**

Ans: In java, each object when created gets a memory space from a heap. When an object is destroyed by a garbage collector, the space allocated to it from the heap is re-allocated to the heap and becomes available for any new objects.

#### **Q86. How can we find the actual size of an object on the heap?**

Ans: In java, there is no way to find out the exact size of an object on the heap.

#### **Q87. Which of the following classes will have more memory allocated?**

**Class A: Three methods, four variables, no object**

**Class B: Five methods, three variables, no object**

Ans: Memory isn't allocated before creation of objects. Since for both classes, there are no objects created so no memory is allocated on heap for any class.

#### **Q88. What happens if an exception is not handled in a program?**

Ans: If an exception is not handled in a program using try catch blocks, program gets aborted and no statement executes after the statement which caused exception throwing.

#### **Q89. I have multiple constructors defined in a class. Is it possible to call a constructor from another constructor's body?**

Ans: If a class has multiple constructors, it's possible to call one constructor from the body of another one using **this()**.

#### **Q90. What's meant by anonymous class?**

Ans: An anonymous class is a class defined without any name in a single line of code using new keyword.

For example, in below code we have defined an anonymous class in one line of code:

```
1 public java.util.Enumeration testMethod()
2
3 {
```

```
4
5 return new java.util.Enumeration()
6
7 {
8
9 @Override
10
11 public boolean hasMoreElements()
12
13 {
14
15 // TODO Auto-generated method stub
16
17 return false;
18
19 }
20
21 @Override
22
23 public Object nextElement()
24
25 {
26
27 // TODO Auto-generated method stub
28
29 return null;
30
31 }
32
33 }
```

### **Q91. Is there a way to increase the size of an array after its declaration?**

Ans: Arrays are static and once we have specified its size, we can't change it. If we want to use such collections where we may require a change of size ( no of items), we should prefer vector over array.

### **Q92. If an application has multiple classes in it, is it okay to have a main method in more than one class?**

Ans: If there is main method in more than one classes in a java application, it won't cause any issue as entry point for any application will be a specific class and code will start from the main method of that particular class only.

### **Q93. I want to persist data of objects for later use. What's the best approach to do so?**

Ans: The best way to persist data for future use is to use the concept of serialization.

### **Q95. String and StringBuffer both represent String objects. Can we compare String and StringBuffer in Java?**

Ans: Although String and StringBuffer both represent String objects, we can't compare them with each other and if we try to compare them, we get an error.

### **Q96. Which API is provided by Java for operations on set of objects?**

Ans: Java provides a Collection API which provides many useful methods which can be applied on a set of objects. Some of the important classes provided by Collection API include ArrayList, HashMap, TreeSet and TreeMap.

### **Q97. Can we cast any other type to Boolean Type with type casting?**

Ans: No, we can neither cast any other primitive type to Boolean data type nor can cast Boolean data type to any other primitive data type.

### **Q98. Can we use different return types for methods when overridden?**

Ans: The basic requirement of method overriding in Java is that the overridden method should have same name, and parameters. But a method can be overridden with a different return type as long as the new return type extends the original.

For example , method is returning a reference type.

```
1 Class B extends A{
2
3 A method(int x){
4
5 //original method
6
7 }
8
9 B method(int x){
10 //overridden method
11 }
12
13 }
14
15 }
```

### **Q99. What's the base class of all exception classes?**

Ans: In Java, **Java.Lang.throwable** is the super class of all exception classes and all exception classes are derived from this base class.

### **Q100. What's the order of call of constructors in inheritance?**

Ans: In case of inheritance, when a new object of a derived class is created, first the constructor of the super class is invoked and then the constructor of the derived class is invoked.

# TOP HTML5 INTERVIEW QUESTIONS

HTML	HTML5
DOCTYPE is much longer	DOCTYPE is required to enable standards mode for HTML documents. <!DOCTYPE html>
Audio and Video are not part of HTML4 specification	Audio and Videos are integral part of HTML5 specifications e.g. <audio> and <video> tags
Vector Graphics is possible with the help of technologies such as VML, Silverlight, Flash etc.	Vector graphics is integral part of HTML5 e.g. <b>SVG</b> and canvas
It is almost impossible to get true GeoLocation of user browsing any website especially if it comes to mobile devices.	JS GeoLocation API in HTML5 helps identify location of user browsing any website (provided user allows it)
Browser cache can be used as temporary storage.	Application Cache, Web SQL database and Web storage is available as client side storage. Accessible using JavaScript interface in HTML5 compliant browsers.
Works with all old browsers	Most of modern browser have started supporting HTML5 specification e.g. Firefox, Mozilla, Opera, Chrome, Safari etc.
Does not allow JavaScript to run in browser. JS runs in same thread as browser interface.	Allows JavaScript to run in background. This is possible due to JS Web worker API in HTML5

## What's new HTML 5 DocType and Charset?

Normally for HTML files **first line of code** is DocType which basically tells browser about **specific version of HTML**. HTML5 is now not subset of SGML. As compared to previous version/standards of HTML, DocType is simplified as follows:

**<!doctype html>**

And HTML 5 uses UTF-8 encoding as follows:

**<meta charset="UTF-8">**

UTF-8 (U from Universal Character Set + Transformation Format—8-bit)

## 2. How can we embed Audio in HTML5?

HTML 5 comes with a standard way of embedding audio files as previously we don't have any such support on a web page. **Supported audio formats are MP3, Wav and Ogg**. Below is the most simple way to embed an audio file on a web page.

```
<audio controls>
 <source src="jamshed.mp3" type="audio/mpeg">
 Your browser does 'nt support audio embedding feature.
</audio>
```



In above code, **src** value can be relative as well as absolute URL.

We can also use multiple **<source>** elements pointing to different audio files. There are more new attributes for **<audio>** tag other than **src** as below:

- **controls** – it adds controls such as volume, play and pause.
- **autoplay** – it's a boolean value which specifies that audio will start playing once it's ready.
- **loop** – it's also a boolean value which specifies looping (means it automatically start playing after it ends).
- **preload** – *auto*, *metadata* and *none* are the possible values for this attribute.
  - *auto* means plays as it loaded.
  - *metadata* displays audio file's associated data
  - *none* means not pre-loaded.

### 3. How can we embed Video in HTML 5?

Same like audio, HTML 5 defined standard way of embedding video files which was not supported in previous versions/standards. Supported video formats are **MP4**, **WebM** and **Ogg**. Please look into below sample code.

```
<video width="450" height="340" controls>
 <source src="jamshed.mp4" type="video/mp4">
 Your browser does 'nt support video embedding feature.
</video>
```

Same like **<audio>**, **<video>** tag has associated optional attributes as controls, autoplay, preload, loop, poster, width, height and other global attributes etc. Controls, loop, preload and autoplay are already explained above. Others are explained below:

- **poster** – it's basically a URL of the image that needs to display until video get started.
- **width** – video player width
- **height** – video player's height

### 4. What are the new media element in HTML 5 other than audio and video?

HTML 5 has strong support for media. Other than audio and video tags, it comes with the following tags:

**<embed> Tag:** **<embed>** acts as a container for external application or some interactive content such as a plug-in. Special about **<embed>** is that it doesn't have a closing tag as we can see below:

```
<embed type="video/quicktime" src="Fishing.mov">
```

**<source> Tag:** **<source>** is helpful for multiple media sources for audio and video.

```
<video width="450" height="340" controls>
 <source src="jamshed.mp4" type="video/mp4">
 <source src="jamshed.ogg" type="video/ogg">
</video>
```

**<track> Tag:** **<track>** defines text track for media like subtitles as:

```
<video width="450" height="340" controls>
 <source src="jamshed.mp4" type="video/mp4">
 <source src="jamshed.ogg" type="video/ogg">
```

```
<track kind="subtitles" label="English" src="jamshed_en.vtt" srclang="en" default></track>
<track kind="subtitles" label="Arabic" src="jamshed_ar.vtt" srclang="ar"></track>
</video>
```

## 5. What is the usage of canvas Element in HTML 5?

<canvas> is an element in HTML5 which we can use to draw graphics with the help of scripting (which is most probably JavaScript).

This element behaves like a container for graphics and rest of things will be done by scripting. We can draw images, graphs and a bit of animations etc using <canvas> element.

```
<canvas id="canvas1" width="300" height="100">
</canvas>
```



As canvas is considered to be the most exciting feature in HTML5, for detailed practical examples on canvas development, you can follow:

- [HTML5 Canvas Tutorials](#)
- [All about Canvas](#)

**Are you going to appear for a Web Development Interview?**  
ASP.NET | ASP.NET MVC | HTML5 | jQuery | AJAX

**Take these** **Free Online Tests** **to see how you do...**

## 6. What are the different types of storage in HTML 5?

HTML 5 has the capability to store data locally. Previously it was done with the help of cookies. Exciting thing about this storage is that its fast as well as secure.

There are two different objects which can be used to store data.

- **localStorage** object stores data for a longer period of time even if the browser is closed.
- **sessionStorage** object stores data for a specific session.

## 7. What are the new Form Elements introduced in HTML 5?

There are a number of new form elements has been introduced in HTML 5 as follows:

- **datalist** provides functionality for **auto-complete feature.** drop down menu
- **datetime** facilitate **selecting a datetime along with Time Zone.**
- **output** represents the **result of a calculation.**
- **keygen** generates a **key-pair field in a form to implement secure authentication.**
- **date** is an input field for date and applies validation accordingly.
- **month** for selecting a month and year in a form input field.
- **week** for selecting a week and year in an input field.
- **time** is an input field for selecting time i.e. Hours:Minutes: AM/PM. For example, 10:30 AM.
- **color** is an input field for color.
- **number** that only allows numeric values.
- **range** is an **input field for selecting value within a specified range.**
- **email** is **input field for email** with standard email validations.
- **url** is for an URL(Uniform Resource Locator) and validated accordingly.

## 8. What are the deprecated Elements in HTML5 from HTML4?

Elements that are deprecated from HTML 4 to HTML 5 are:

- **frame**
- **frameset**
- **noframe**
- **applet**
- **big**
- **center**
- **basefront**

## 9. What are the new APIs provided by HTML 5 standard?

HTML 5 standard comes with a number of new APIs. Few of it are as follows:

- **Media API**
- **Text Track API**
- **Application Cache API**
- **User Interaction**
- **Data Transfer API**
- **Command API**
- **Constraint Validation API**
- **History API**
- and many more....

## 10. What is the difference between HTML 5 Application Cache and regular HTML Browser Cache?

One of the key feature of HTML 5 is “Application Cache” that enables us to make an offline version of a web application. It allows to fetch few or all of website contents such as HTML files, CSS, images, javascript etc locally. This feature speeds up the site performance. This is achieved with the help of a manifest file defined as follows:

```
<!doctype html>
<html manifest="example.appcache">
.....
</html>
```

As compared with traditional browser caching, Its not compulsory for the user to visit website contents to be cached.

## 11. What is the major improvement with HTML5 in reference to Flash?

Flash is not supported by major mobile devices such as iPad, iPhone and universal android applications. Those mobile devices have lack of support for installing flash plugins. HTML5 is supported by all the devices, apps and browser including Apple and Android products. Compared to Flash, HTML5 is very secured and protected. That eliminates major concerns that we have seen with Flash.

An average recruiter has 6 seconds to check your resume... will you be noticed?

[Check Premium Resumes](#)



## Top 24 C++ Interview questions

### 1) Explain what is a class in C++?

A class in C++ can be defined as a collection of function and related data under a single name. It is a blueprint of objects. A C++ program can consist of any number of classes.

### 2) How can you specify a class in C++?

By using the keyword class followed by identifier (name of class) you can specify the class in C++.

Inside curly brackets, body of the class is defined. It is terminated by semi-colon in the end.  
[crayon-54f5d09240752336128978/]

### 3) Explain what is the use of void main () in C++ language?

To run the C++ application it involves two steps, the first step is a compilation where conversion of C++ code to object code take place. While second step includes linking, where combining of object code from the programmer and from libraries takes place. This function is operated by main () in C++ language.

### 4) Explain what is C++ objects?

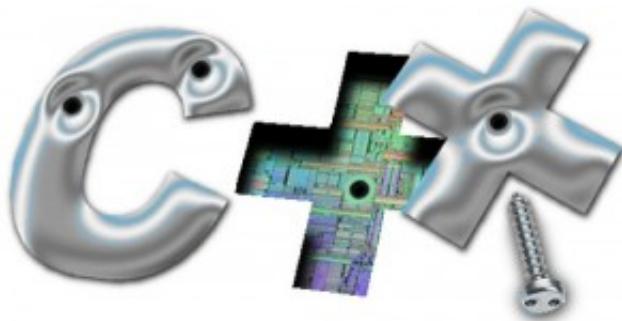
Class gives blueprints for object, so basically an object is created from a class or in other words an object is an instance of a class. The data and functions are bundled together as a self-contained unit called an object. Here, in the example A and B is the Object.

For example,

[crayon-54f5d0924075a473658409/]

### 5) Explain what are the characteristics of Class Members in C++?

- Data and Functions are members in C++,
- Within the class definition, data members and methods must be declared
- Within a class, a member cannot be re-declare
- Other than in the class definition, no member can be added elsewhere



## 6) Explain what is Member Functions in Classes?

The member function regulates the behaviour of the class. It provides a definition for supporting various operations on data held in the form of an object.

## 7) Define basic type of variable used for a different condition in C++?

The variable used for a different condition in C++ are

- Bool: Variable to store boolean values (true or false)
- Char: Variable to store character types
- int : Variable with integral values
- float and double: Types of variables with large and floating point values

## 8) What is namespace std; and what is consists of?

Namespace std; defines your standard C++ library, it consists of classes, objects and functions of the standard C++ library. You can specify the library by using namespace std or std: : throughout the code. Namespace is used to differentiate the same functions in a library by defining the name.

## 9) Explain what is Loop function? What are different types of Loops?

In any programming language, to execute a set of statements repeatedly until a particular condition is satisfied Loop function is used. The loop statement is kept under the curly braces {} referred as Loop body.

In C++ language, three types of loops are used

- While loop
- For loop
- Do-while loop

## 10) Explain how functions are classified in C++ ?

In C++ functions are classified as

- Return type
- Function Name
- Parameters
- Function body

### **11) Explain what are Access specifiers in C++ class? What are the types?**

Access specifiers determine the access rights for the statements or functions that follow it until the end of class or another specifier is included. Access specifiers decide how the members of the class can be accessed. There are three types of specifiers

- Private
- Public
- Protected

### **12) Explain what are Operators and explain with an example?**

Operators are specific operands in C++ that is used to perform specific operations to obtain a result. The different types of operators available for C++ are Assignment Operator, Compound Assignment Operator, Arithmetic Operator, Increment Operator and so on.

For example arithmetic operators, you want to add two values a+b

[crayon-54f5d0924075f933734421/]

It will give the output as 31 when you run the command

### **13) What is the C-style character string?**

The string is actually a one-dimensional array of characters that is terminated by a null character '\0'.

For example, to type hello word

[crayon-54f5d09240762247319420/]

On executing this code it will give the result like? Greeting message: Hello

### **14) Explain what is a reference variable in C++?**

A reference variable is just like a pointer with few differences. It is declared using & Operator. In other words, reference is another name for an already existing variable.

### **15) Explain what is Polymorphism in C++?**

Polymorphism in C++ is the ability to call different functions by using only one type of the function call. Polymorphism is referred to codes, operations or objects that behave differently in a different context.

For example, the addition function can be used in many contexts like

- 5+5 ? Integer addition
- Medical+Internship ? The same ( + ) operator can be used with different meaning with strings
- 3.14 + 2.27 ? The same ( + ) operator can be used for floating point addition

## **16) Explain what is data abstraction in C++?**

Data abstraction is a technique to provide essential information to the outside world while hiding the background details. Here in below example you don't have to understand how cout display the text "Hello guru99" on the user screen and at the same time implementation of cout is free to change

For example,

[crayon-54f5d09240765257858135/]

## **17) Explain what is C++ exceptional handling?**

The problem that arises during execution of a program is referred as exceptional handling. The exceptional handling in C++ is done by three keywords.

- Try: It identifies a block of code for which particular exceptions will be activated
- Catch: The catch keyword indicates the catching of an exception by an exception handler at the place in a program
- Throw: When a problem exists while running the code, the program throws an exception

## **18) Explain what is data encapsulation in C++?**

Encapsulation is an object oriented programming concept (oops) which binds together the data and functions. It is also referred as data hiding mechanism.

## **19) Mention what are the types of Member Functions?**

The types of member functions are

- Simple functions
- Static functions
- Const functions
- Inline functions
- Friend functions

## **20) Mention what are the decision making statements in C++? Explain if statement with an example?**

The decision making statements in C++ are

- if statement

- switch statement
- conditional operator
- goto statement

For example, we want to implement if condition in C++

[crayon-54f5d09240768768009936/]

## 21) Explain what is multi-threading in C++?

To run two or more programs simultaneously multi-threading is useful. There are two types of

- Process-based: It handles the concurrent execution of the program
- Thread-based: It deals with the concurrent execution of pieces of the same program

## 22) Explain what is upcasting in C++?

Upcasting is the act of converting a sub class references or pointer into its super class reference or pointer is called upcasting.

## 23) Explain what is pre-processor in C++?

Pre-processors are the directives, which give instruction to the compiler to pre-process the information before actual compilation starts.

## 24) Explain what is COPY CONSTRUCTOR and what is it used for?

COPY CONSTRUCTOR is a technique that accepts an object of the same class and copies its data member to an object on the left part of the assignment.

Guru99 provides [FREE ONLINE TUTORIAL](#) on Various courses like

[PHP](#)

[Java](#)

[Linux](#)

[Apache](#)

[Perl](#)

[SQL](#)

[VB Script](#)

[JavaScript](#)

[Accounting](#)

[Ethical Hacking](#)

[Cloud Computing](#)

[Jmeter](#)

[Manual Testing](#)

[QTP](#)

[Selenium](#)

[Test Management](#)

[Load Runner](#)

[Quality Center](#)

[Mobile Testing](#)

[Live Selenium Project](#)

[Enterprise Testing](#)

[Live Testing Project](#)

[Sap & All Modules](#)



---

Copyrighted Material

# Top 50 CSS (Cascading Style Sheet) Interview Questions

## 1. What is CSS?

The full form of CSS is **Cascading Style Sheets**. It is a **styling language which is simple enough for HTML elements**. It is popular in web designing, and its application is common in XHTML also.

## 2. What is the origin of CSS?

**Standard Generalized Markup Language** marked the beginning of style sheets in 1980s.

## 3. What are the different variations of CSS?

The variations for CSS are:

- **CSS 1**
- **CSS 2**
- **CSS 2.1**
- **CSS 3**
- **CSS 4**

## 4. What are the limitations of CSS?

Limitations are:

- Ascending by selectors is not possible
- **Limitations of vertical control**
- No expressions
- **No column declaration**
- Pseudo-class not controlled by dynamic behavior
- **Rules, styles, targeting specific text not possible**

## 5. What are the advantages of CSS?

Advantages are:

- **Bandwidth**
- **Site-wide consistency**
- **Page reformatting**
- **Accessibility**
- **Content separated from presentation**

```

1 /* GENERAL STYLES
2 *-----*/
3 html, body, form, fieldset, img, img a {
4 margin: 0;
5 padding: 0;
6 border: 0;
7 }
8 body {
9 color: #414141;
10 background: url(../images/bg.jpg) repeat-x #ebe8df;
11 font-family: Arial, Helvetica, sans-serif;
12 line-height: 128%;
13 font-size: 12px;
14 }
15
16 a:link, a:visited {
17 color: #685966;
18 text-decoration: underline;
19 }
20 a:hover {
21 color: #2b212c;
22 }
23 .article_separator {
24 line-height: 5px;
25 height: 5px;
26 font-size: 5px;
27 }
28 /* SITE WIDTH
29 *-----*/
30 .rht_container {
31 width: 1020px;
32 margin: 0 auto;
33 margin-top: 25px;

```

## 6. What are CSS frameworks?

It is a pre-planned libraries, which allows easier and more standards-compliant webpage styling, using CSS language.

## 7. How block elements can be centered with CSS1?

Block level elements can be centered by:

The margin-left and margin-right properties can be set to some explicit value:

```

1 BODY {
2
3 width: 40em;
4
5 background: fluorescent;
6
7 }
8
9 P {
10
11 width: 30em;
12
13 margin-right: auto;
14
15 margin-left: auto
16
17 }

```

In this case, the left and right margins will be each, five ems wide since they split up the ten ems left over from (40em-30em). It was unnecessary for setting up an explicit width for the BODY element; it was done here for simplicity.

## 8. Who maintains the CSS specifications?

World Wide Web Consortium maintains the CSS specifications.

## 9. In how many ways can a CSS be integrated as a web page?

CSS can be integrated in three ways:

- **Inline:** Style attribute can be used to have CSS applied to HTML elements.
- **Embedded:** The Head element can have a Style element within which the code can be placed.
- **Linked/ Imported:** CSS can be placed in an external file and linked via link element.

## 10. What benefits and demerits do External Style Sheets have?

Benefits:

- One file can be used to control multiple documents having different styles.
- Multiple HTML elements can have many documents, which can have classes.
- To group styles in composite situations, methods as selector and grouping are used.

Demerits:

- Extra download is needed to import documents having style information.
- To render the document, the external style sheet should be loaded.
- Not practical for small style definitions.

## 11. Discuss the merits and demerits of Embedded Style Sheets?

Merits of Embedded Style Sheets:

- Multiple tag types can be created in a single document.
- Styles, in complex situations, can be applied by using Selector and Grouping methods.
- Extra download is unnecessary.

Demerits of Embedded Style Sheets:

- Multiple documents cannot be controlled.

## 12. What does CSS selector mean?

A string equivalent of HTML elements by which declarations or a set of it, is declared and is a link that can be referred for linking HTML and Style sheet is CSS selector.

### **13. Enlist the media types CSS allows?**

The design and customization of documents are rendered by media. By applying media control over the external style sheets, they can be retrieved and used by loading it from the network.

### **14. Differentiate logical tags from physical tags?**

- While physical tags are also referred to as presentational mark-up, logical tags are useless for appearances.
- Physical tags are newer versions while logical tags are old and concentrate on content.

### **15. Differentiate Style Sheet concept from HTML?**

While HTML provides easy structure method, it lacks styling, unlike Style sheets. Moreover, style sheets have better browser capabilities and formatting options.

### **16. Describe ‘ruleset’?**

Ruleset: Selectors can be attached to other selectors to be identified by ruleset.

It has two parts:

- Selector, e.g. R and
- declaration {text-indent: 11pt}

### **17. Comment on the Case-sensitivity of CSS?**

Although, there are no case-sensitivity of CSS, nevertheless font families, URL's of images, etc is. Only when XML declarations along with XHTML DOCTYPE are being used on the page, CSS is case -sensitive.

### **18. Define Declaration block?**

A catalog of directions within braces consisting of property, colon and value is called declaration block.

e.g.: [property 1: value 3]

### **19. Enlist the various fonts' attributes?**

They are:

- Font-style
- Font-variant
- Font-weight
- Font-size/line-height
- Font-family
- Caption
- Icon

## **20. Why is it easy to insert a file by importing it?**

Importing enables combining external sheets to be inserted in many sheets. Different files and sheets can be used to have different functions. Syntax:

@import notation, used with <Style> tag.

## **21. What is the usage of Class selector?**

Selectors that are unique to a specific style, are called CLASS selectors. Declaration of style and association with HTML can be made through this. Syntax:

Classname

it can be A-Z, a-z or digits.

.top { font: 14em ;}, class selector

<Body class= “top”> this class is associated with element </body>

## **22. Differentiate Class selector from ID selector?**

While an overall block is given to class selector, ID selector prefers only a single element differing from other elements.

## **23. Can more than one declaration be added in CSS?**

Yes, it can be achieved by using a semicolon.

## **24. What is Pseudo-elements?**

Pseudo-elements are used to add special effects to some selectors. CSS is used to apply styles in HTML mark-up. In some cases when extra mark-up or styling is not possible for the document, then there is a feature available in CSS known as pseudo-elements. It will allow extra mark-up to the document without disturbing the actual document.

## **25. How to overrule underlining Hyperlinks?**

Control statements and external style sheets are used to overrule underlining Hyperlinks.

E.g.:

```
1 B {
2
3 text-decoration: none;
4
5 }
6
7 <B href="career.html" style="text-decoration: none">link text
```

## **26. What happens if 100% width is used along with floats all across the page?**

While making the float declaration, 1 pixel is added every time it is used in the form of the border, and even more float is allowed thereafter.

## **27. Can default property value be restored through CSS? If yes, how?**

In CSS, you cannot revert back to old values due to lack of default values. The property can be re-declared to get the default property.

## **28. Enlist the various Media types used?**

Different media has different properties as they are case insensitive.

They are:

- Aural – for sound synthesizers and speech
- Print – gives a preview of the content when printed
- Projection- projects the CSS on projectors.
- Handheld- uses handheld devices.
- Screen- computers and laptop screens.

## **29. What is CSS Box Model and what are its elements?**

This box defines design and layout of elements of CSS. The elements are:

**Margin:** the top most layer, the overall structure is shown

**Border:** the padding and content option with a border around it is shown. Background color affects the border.

**Padding:** Space is shown. Background color affects the border.

**Content:** Actual content is shown.

## **30. What is contextual selector?**

Selector used to select special occurrences of an element is called contextual selector. A space separates the individual selectors. Only the last element of the pattern is addressed in this kind of selector. For e.g.: TD P TEXT {color: blue}

## **31. Compare RGB values with Hexadecimal color codes?**

A color can be specified in two ways:

- A color is represented by 6 characters i.e. hexadecimal color coding. It is a combination of numbers and letters and is preceded by #. e.g.: g {color: #00cjfi}
- A color is represented by a mixture of red, green and blue. The value of a color can also be specified. e.g.: rgb(r,g,b): In this type the values can be in between the integers 0 and 255. rgb(r%,g%,b%): red, green and blue percentage is shown.

## 32. Define Image sprites with context to CSS?

When a set of images is collaborated into one image, it is known as ‘Image Sprites’. As the loading every image on a webpage consumes time, using image sprites lessens the time taken and gives information quickly.

CSS coding:

```
1 img.add { width: 60px; height: 55px; background: url (image.god) 0 0; }
```

In this case, only the part needed is used. The user can save substantial margin and time through this.

## 33. Compare Grouping and Nesting in CSS?

Grouping: Selectors can be grouped having the same values of property and the code be reduced.

E.g.:

```
1 h1 {
2
3 color: blue;
4
5 }
6
7 h2 {
8
9 color: blue;
10
11 }
12
13 p {
14
15 color: blue;
16
17 }
```

It can be seen from the code that every element shares the same property. Rewriting can be avoided by writing each selector separated by a comma.

Nesting: Specifying a selector within a selector is called nesting.

```
1 P
2
3 {
4
5 color: red;
6
7 text-align: left;
8
9 }
10
11 .marked
```

```
12
13 {
14
15 background-color: blue;
16
17 }
18
19 .marked p
20
21 {
22
23 color: green;
24 }
25
```

#### **34. How can the dimension be defined of an element?**

Dimension properties can be defined by:

- Height
- Max-height
- Max-width
- Min-height
- Min-width
- Width

#### **35. Define float property of CSS?**

By float property, the image can be moved to the right or the left along with the text to be wrapped around it. Elements before this property is applied do not change their properties.

#### **36. How does Z index function?**

Overlapping may occur while using CSS for positioning HTML elements. Z index helps in specifying the overlapping element. It is a number which can be positive or negative, the default value being zero.

#### **37. What is graceful degradation?**

In case the component fails, it will continue to work properly in the presence of a graceful degradation. The latest browser application is used when a webpage is designed. As it is not available to everyone, there is a basic functionality, which enables its use to a wider audience. In case the image is unavailable for viewing, text is shown with the alt tag.

### **38. What is progressive enhancement?**

It's an alternative to graceful degradation, which concentrates on the matter of the web. The functionality is same, but it provides an extra edge to users having the latest bandwidth. It has been into prominent use recently with mobile internet connections expanding their base.

### **39. How can backward compatibility be designed in CSS?**

HTML sheet methods is collaborated with CSS and used accordingly.

### **40. How can the gap under the image be removed?**

As images being inline elements are treated same as texts, so there is a gap left, which can be removed by:  
**CSS**

```
1 img { display: block ; }
```

### **41. Why is @import only at the top?**

@import is preferred only at the top, to avoid any overriding rules. Generally, ranking order is followed in most programming languages such as Java, Modula, etc. In C, the # is a prominent example of a @import being at the top.

### **42. Which among the following is more precedent: CSS properties or HTML procedures?**

CSS is more precedent over HTML procedures. Browsers, which do not have CSS support, display HTML attributes.

### **43. What is Inline style?**

The Inline style in a CSS is used to add up styling to individual HTML elements.

### **44. How comments can be added in CSS?**

The comments in CSS can be added with /\* and \*/.

### **45. Define Attribute Selector?**

It is defined by a set of elements, value and its parts.

### **46. Define property?**

A style, that helps in influencing CSS. E.g. FONT. They have corresponding values or properties within them, like FONT has different style like bold, italic etc.

#### **47. What is Alternate Style Sheet?**

Alternate Style Sheets allows the user to select the style in which the page is displayed using the view>page style menu. Through Alternate Style Sheet, user can see a multiple version of the page on their needs and preferences.

#### **48. Are quotes mandatory in URL's?**

Quotes are optional in URL's, and it can be single or double.

#### **49. What is at-rule?**

Rule, which is applicable in the entire sheet and not partly, is known as at-rule. It is preceded by @ followed by A-Z, a-z or 0-9.

#### **50. How can CSS be cascaded to mix with user's personal sheet?**

Properties can be a set in recommended places and the document modified for CSS to mix with user's personal sheet.

An average recruiter has 6 seconds to check your resume... will you be noticed?

[Check Premium Resumes](#)



## Top 50 Data Structure Interview Questions

### 1) What is data structure?

Data structure refers to the way data is organized and manipulated. It seeks to find ways to make data access more efficient. When dealing with data structure, we not only focus on one piece of data, but rather different set of data and how they can relate to one another in an organized manner.

### 2) Differentiate file structure from storage structure.

Basically, the key difference is the memory area that is being accessed. When dealing with the structure that resides the main memory of the computer system, this is referred to as storage structure. When dealing with an auxiliary structure, we refer to it as file structure.

### 3) When is a binary search best applied?

A binary search is an algorithm that is best applied to search a list when the elements are already in order or sorted. The list is search starting in the middle, such that if that middle value is not the target search key, it will check to see if it will continue the search on the lower half of the list or the higher half. The split and search will then continue in the same manner.

### 4) What is a linked list?

A linked list is a sequence of nodes in which each node is connected to the node following it. This forms a chain-like link of data storage.

### 5) How do you reference all the elements in a one-dimension array?

To do this, an indexed loop is used, such that the counter runs from 0 to the array size minus one. In this manner, we are able to reference all the elements in sequence by using the loop counter as the array subscript.

### 6) In what areas do data structures applied?

Data structure is important in almost every aspect where data is involved. In general, algorithms

that involve efficient data structure is applied in the following areas: numerical analysis, operating system, A.I., compiler design, database management, graphics, and statistical analysis, to name a few.

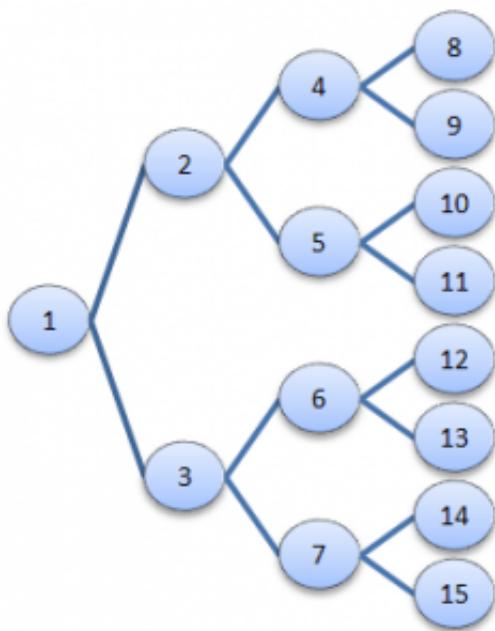
## 7) What is LIFO?

LIFO is short for Last In First Out, and refers to how data is accessed, stored and retrieved. Using this scheme, data that was stored last, should be the one to be extracted first. This also means that in order to gain access to the first data, all the other data that was stored before this first data must first be retrieved and extracted.

## 8 ) What is a queue?

A queue is a data structure that can simulates a list or stream of data. In this structure, new elements are inserted at one end and existing elements are removed from the other end.

## 9) What are binary trees?



A binary tree is one type of data structure that has two nodes, a left node and a right node. In programming, binary trees are actually an extension of the linked list structures.

## 10) Which data structure is applied when dealing with a recursive function?

Recursion, which is basically a function that calls itself based on a terminating condition, makes use of the stack. Using LIFO, a call to a recursive function saves the return address so that it knows how to return to the calling function after the call terminates.

## 11) What is a stack?

---

A stack is a data structure in which only the top element can be accessed. As data is stored in the stack, each data is pushed downward, leaving the most recently added data on top.

## 12) Explain Binary Search Tree

A binary search tree stores data in such a way that they can be retrieved very efficiently. The left subtree contains nodes whose keys are less than the node's key value, while the right subtree contains nodes whose keys are greater than or equal to the node's key value. Moreover, both subtrees are also binary search trees.

## 13) What are multidimensional arrays?

Multidimensional arrays make use of multiple indexes to store data. It is useful when storing data that cannot be represented using a single dimensional indexing, such as data representation in a board game, tables with data stored in more than one column.

## 14) Are linked lists considered linear or non-linear data structure?

It actually depends on where you intend to apply linked lists. If you based it on storage, a linked list is considered non-linear. On the other hand, if you based it on access strategies, then a linked list is considered linear.

## 15) How does dynamic memory allocation help in managing data?

Aside from being able to store simple structured data types, dynamic memory allocation can combine separately allocated structured blocks to form composite structures that expand and contract as needed.

## 16) What is FIFO?

FIFO is short for First-in, First-out, and is used to represent how data is accessed in a queue. Data has been inserted into the queue list the longest is the one that is removed first.

## 17) What is an ordered list?

An ordered list is a list in which each node's position in the list is determined by the value of its key component, so that the key values form an increasing sequence, as the list is traversed.

## 18) What is merge sort?

Merge sort takes a divide-and-conquer approach to sorting data. In a sequence of data, adjacent ones are merged and sorted to create bigger sorted lists. These sorted lists are then merged again to form an even bigger sorted list, which continues until you have one single sorted list.

## 19) Differentiate NULL and VOID.

---

Null is actually a value, whereas Void is a data type identifier. A variable that is given a Null value simply indicates an empty value. Void is used to identify pointers as having no initial size.

## **20) What is the primary advantage of a linked list?**

A linked list is a **very ideal data structure because it can be modified easily**. This means that modifying a linked list works regardless of how many elements are in the list.

## **21) What is the difference between a PUSH and a POP?**

Pushing and popping applies to the way data is stored and retrieved in a stack. A push denotes data being added to it, meaning data is being “pushed” into the stack. On the other hand, a pop denotes data retrieval, and in particular refers to the topmost data being accessed.

## **22) What is a linear search?**

A linear search refers to the way a target key is being searched in a sequential data structure. Using this method, each element in the list is checked and compared against the target key, and is repeated until found or if the end of the list has been reached.

## **23) How does variable declaration affect memory allocation?**

The amount of memory to be allocated or reserved would depend on the data type of the variable being declared. For example, if a variable is declared to be of integer type, then 32 bits of memory storage will be reserved for that variable.

## **24) What is the advantage of the heap over a stack?**

Basically, the heap is more flexible than the stack. That's because memory space for the heap can be dynamically allocated and de-allocated as needed. However, memory of the heap can at times be slower when compared to that stack.

## **25) What is a postfix expression?**

A postfix expression is an expression in which each operator follows its operands. The advantage of this form is that there is no need to group sub-expressions in parentheses or to consider operator precedence.

## **26) What is Data abstraction?**

Data abstraction is a powerful tool for breaking down complex data problems into manageable chunks. This is applied by initially specifying the data objects involved and the operations to be performed on these data objects without being overly concerned with how the data objects will be represented and stored in memory.

## **27) How do you insert a new item in a binary search tree?**

---

Assuming that the data to be inserted is a unique value (that is, not an existing entry in the tree), check first if the tree is empty. If it's empty, just insert the new item in the root node. If it's not empty, refer to the new item's key. If it's smaller than the root's key, insert it into the root's left subtree, otherwise, insert it into the root's right subtree.

## **28) How does a selection sort work for an array?**

The selection sort is a **fairly intuitive sorting algorithm**, though not necessarily efficient. To perform this, the smallest element is first located and switched with the element at subscript zero, thereby **placing the smallest element in the first position**. The smallest element remaining in the subarray is then located next with subscripts 1 through n-1 and switched with the element at subscript 1, thereby placing the second smallest element in the second position. The steps are repeated in the same manner till the last element.

## **29) How do signed and unsigned numbers affect memory?**

In the case of signed numbers, the first bit is used to indicate whether positive or negative, which leaves you with one bit short. With unsigned numbers, you have all bits available for that number. The effect is best seen in the number range (unsigned 8 bit number has a range 0-255, while 8-bit signed number has a range -128 to +127).

## **30) What is the minimum number of nodes that a binary tree can have?**

A binary tree can have a minimum of zero nodes, which occurs when the nodes have NULL values. Furthermore, a binary tree can also have 1 or 2 nodes.

## **31) What are dynamic data structures?**

Dynamic data structures are structures that expand and contract as a program runs. It provides a flexible means of manipulating data because it can adjust according to the size of the data.

## **32) In what data structures are pointers applied?**

Pointers that are used in linked list have various applications in data structure. Data structures that make use of this concept include the Stack, Queue, Linked List and Binary Tree.

## **33) Do all declaration statements result in a fixed reservation in memory?**

Most declarations do, with the exemption of pointers. Pointer declaration does not allocate memory for data, but for the address of the pointer variable. Actual memory allocation for the data comes during run-time.

## **34) What are ARRAYS?**

When dealing with arrays, **data is stored and retrieved using an index** that actually refers to the element number in the data sequence. This means that data can be accessed in any order. In

programming, an array is declared as a variable having a number of indexed elements.

### 35) What is the minimum number of queues needed when implementing a priority queue?

The minimum number of queues needed in this case is two. One queue is intended for sorting priorities while the other queue is intended for actual storage of data.

### 36) Which sorting algorithm is considered the fastest?

There are many types of sorting algorithms: quick sort, bubble sort, balloon sort, radix sort, merge sort, etc. Not one can be considered the fastest because each algorithm is designed for a particular data structure and data set. It would depend on the data set that you would want to sort.

### 37) Differentiate STACK from ARRAY.

Data that is stored in a stack follows a LIFO pattern. This means that data access follows a sequence wherein the last data to be stored will be the first one to be extracted. Arrays, on the other hand, does not follow a particular order and instead can be accessed by referring to the indexed element within the array.

### 38) Give a basic algorithm for searching a binary search tree.

1. if the tree is empty, then the target is not in the tree, end search
2. if the tree is not empty, the target is in the tree
3. check if the target is in the root item
4. if target is not in the root item, check if target is smaller than the root's value
5. if target is smaller than the root's value, search the left subtree
6. else, search the right subtree

### 39) What is a dequeue?

A dequeue is a double-ended queue. This is a structure wherein elements can be inserted or removed from either end.

### 40) What is a bubble sort and how do you perform it?

A bubble sort is one sorting technique that can be applied to data structures such as an array. It works by comparing adjacent elements and exchanges their values if they are out of order. This method lets the smaller values “bubble” to the top of the list, while the larger value sinks to the bottom.

### 41) What are the parts of a linked list?

A linked list typically has two parts: the head and the tail. Between the head and tail lie the actual nodes, with each node being linked in a sequential manner.

**42) How does selection sort work?**

Selection sort works by picking the smallest number from the list and placing it at the front. This process is repeated for the second position towards the end of the list. It is the simplest sort algorithm.

**43) What is a graph?**

A graph is one type of data structure that contains a set of ordered pairs. These ordered pairs are also referred to as edges or arcs, and are used to connect nodes where data can be stored and retrieved.

**44) Differentiate linear from non linear data structure.**

Linear data structure is a structure wherein data elements are adjacent to each other. Examples of linear data structure include arrays, linked lists, stacks and queues. On the other hand, non-linear data structure is a structure wherein each data element can connect to more than two adjacent data elements. Examples of non linear data structure include trees and graphs.

**45) What is an AVL tree?**

An AVL tree is a type of binary search tree that is always in a state of partially balanced. The balance is measured as a difference between the heights of the subtrees from the root. This self-balancing tree was known to be the first data structure to be designed as such.

**46) What are doubly linked lists?**

Doubly linked lists are a special type of linked list wherein traversal across the data elements can be done in both directions. This is made possible by having two links in every node, one that links to the next node and other one that links to the previous node.

**47) What is Huffman's algorithm?**

Huffman's algorithm is associated in creating extended binary trees that has minimum weighted path lengths from the given weights. It makes use of a table that contains frequency of occurrence for each data element.

**48) What is Fibonacci search?**

Fibonacci search is a search algorithm that applies to a sorted array. It makes use of a divide-and-conquer approach that can greatly reduce the time needed in order to reach the target element.

**49) Briefly explain recursive algorithm.**

Recursive algorithm targets a problem by dividing it into smaller, manageable sub-problems.

The output of one recursion after processing one sub-problem becomes the input to the next recursive process.

## 50) How do you search for a target key in a linked list?

To find the target key in a linked list, you have to apply sequential search. Each node is traversed and compared with the target key, and if it is different, then it follows the link to the next node. This traversal continues until either the target key is found or if the last node is reached.

Guru99 provides [FREE ONLINE TUTORIAL](#) on Various courses like

[PHP](#)[Java](#)[Linux](#)[Apache](#)[Perl](#)[SQL](#)[VB Script](#)[JavaScript](#)[Accounting](#)[Ethical Hacking](#)[Cloud Computing](#)[Jmeter](#)[Manual Testing](#)[QTP](#)[Selenium](#)[Test Management](#)[Load Runner](#)[Quality Center](#)[Mobile Testing](#)[Live Selenium Project](#)[Enterprise Testing](#)[Live Testing Project](#)[Sap & All Modules](#)

-----  
Copyrighted Material

## DATA STRUCTURES QUESTIONS:

- Time complexity in Data Structures – O (n<sup>2</sup>)

### Reversing a linked list in Java

```
LinkedList<Character> linkedList = new LinkedList<Character>();
String str = "Replace me with your code!";
```

```
for (int i = 0; i < str.length(); i++)
{
 linkedList.addFirst(str.charAt(i));
 System.out.println(str.charAt(i));
}

//whenever it is time to print the value....
for (char c : linkedList) {
 //Print it out, one character at a time
 System.out.print(c);
}
```

### What is difference between Stack and Queue data structure?

Any way main difference is that Stack is LIFO (Last In First Out) data structure while Queue is a FIFO (First In First Out) data structure.

### How do you find duplicates in array if there is more than one duplicate?

One way of solving this problem is using a [Hashtable or HashMap](#) data structure. In Java if a number already exists in [HashMap](#) then calling get(index) will return number otherwise it return null. This property can be used to insert or update numbers in HashMap.

### What is difference between Singly Linked List and Doubly Linked List data structure?

Main difference between singly linked list and doubly linked list is ability to traverse. In a single linked list, node only points towards next node, and there is no pointer to previous node, which means you cannot traverse back on a singly linked list. On the other hand doubly linked list maintains two pointers, towards next and previous node, which allows you to navigate in both direction in any linked list.

A **hash table** (hash map) is a data structure used to implement an associative array, a structure that can map keys to values.

```
Hashtable balance = new Hashtable();

balance.put("Zara", new Double(3434.34));
balance.put("Mahnaz", new Double(123.22));
```

**ArrayList** is implemented as a resizable array. As more elements are added to ArrayList, its size is increased dynamically. Its elements can be accessed directly by using the get and set methods.

**LinkedList** is actually **luxury extension of an ArrayList**: **LinkedList** is **much more flexible and lets you insert, add and remove elements**

**Depth-first traversal** is further classified by position of the root element with regard to the left and right nodes.

**Integer matrix:** A matrix resembles a table with rows and columns using integer variables.

Leetcode Problem: **Linked List Cycle** : *Given a linked list, determine if it has a cycle in it.*

**Time complexity of different sorting algorithms.** You can go to wiki to see basic idea of them.

Algorithm	Average Time	Worst Time	Space
Bubble sort	$n^2$	$n^2$	1
Selection sort	$n^2$	$n^2$	1
Insertion sort	$n^2$	$n^2$	
Quick sort	$n \log(n)$	$n^2$	
Merge sort	$n \log(n)$	$n \log(n)$	depends

**Graph** related questions mainly focus on depth first search and breath first search.



# TOP 50 EJB

## 1. What is EJB?

A server-side component, which manages the architecture for constricting enterprise applications and managed is called Enterprise JavaBeans(EJB).

## 2. When was EJB developed?

EJB was developed by IBM in 1997.

## 3. Who took over EJB?

EJB was taken over by Sun Microsystems in 1999.

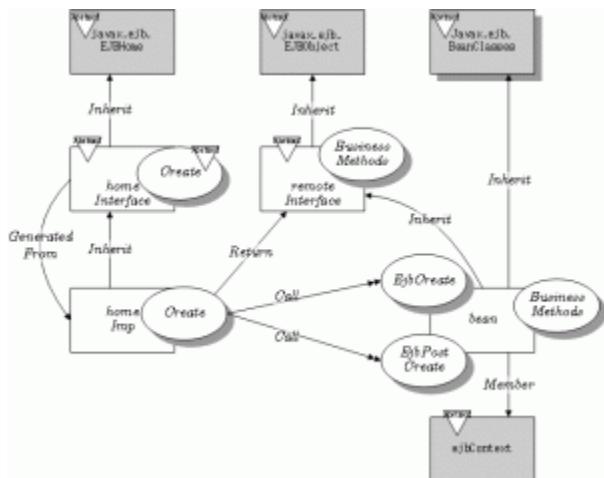
## 4. Enlist the Enterprise Beans types?

They are:

- Session Beans: Expanded as “Stateful”, “Stateless” and “Singleton”, A Remote or Local interface is used to access the EJB files.
- Message Driven Beans (MDB): Asynchronous execution by means of messaging paradigm is supported.

## 5. What were Entity Beans?

Entity Beans were presented in the earlier versions of EJB consisting of persistent data in distributed objects.



EJB

## **6. Enlist the Declarative Transaction types?**

They are:

- MANDATORY:
- REQUIRED
- REQUIRES\_NEW
- SUPPORTS
- NOT\_SUPPORTED
- NEVER

## **7. What are versions of EJB?**

- EJB 3.1
- EJB 3.2
- EJB 3.2 Final Release (2013-05-28)
- EJB 3.1 Final Release (2009-12-10)
- EJB 3.0 Final Release (2006-05-11)
- EJB 2.1, Final Release (2003-11-24)
- EJB 2.0 Final Release (2001-08-22)
- EJB 1.1 Final Release (1999-12-17)
- EJB 1.0 (1998-03-24)

## **8. What is J2EE?**

A collection of synchronized specifications and procedures, which enable solutions regarding deploying, developing supervising multi-tier server-centric applications, is called J2EE.

## **9. Enlist the changes in EJB 2.1?**

The changes made are:

- Message Driven Beans (MDBs): messages are accepted from other sources besides JMS.
- EJB Query Language: New functions had been added.
- Web services supported.
- EJB Timer Service: Mechanism based on an event to invoke EJBs at particular times.
- XML schema
- Message destinations

## **10. Enlist the contents of Container. • Container contains?**

- Security support: Used to configure Deployment Descriptor (DD)
- Persistence support: Used to be persistence in transactions.
- Transaction management support: Used to configure Deployment Descriptor (DD)
- Management of Session: Callback methods such as ejbStore (), ejbLoad () are used in the Developer.
- Management of Life Cycle: Automatic

## **11. Differentiate ‘Stateful Session’ from ‘Entity Bean’ ?**

While both undergo activation and passivation; EB have ejbStore () callback to save state through passivation and ejbLoad () callback to load state through activation. But in case of SS, this is not needed because S.S.B fields are serialized through objects by containers.

## **12. Which is more beneficial: Stateful or Stateless Bean?**

If a conversational state is needed then, Stateful mode is preferred while Stateless paradigm is preferred for a single business process.

## **13. Which is more beneficial: CMP or BMP?**

When “one to one” mapping is involved, and the data is stored persistently in regional database, CMP is preferred. But when no “one to one” mapping is there and data is retrieved from numerous tables having a complex query, Bean Managed Persistence is used.

## **14. How is consistency maintained by Stateful Session through transaction updates ?**

The data consistency is maintained by updating their fields every time a commitment of the transaction is made.

## **15. Is ejbCreate () method mandatory while defining a Session Bean?**

EjbCreate () as being part of the bean’s lifecycle, therefore, it is not mandatory for ejbCreate () method to be present and there will be no errors returned by the compiler.

## **16. Define Context?**

This is a method of binding a name to a specific object by giving an interface like javax.naming.Context.

## **17. Define Initial Context?**

Implementation of available methods in the interface of context such as a context called javax.naming.InitialContext.

## **18. Define SessionContext ?**

An EJBContext object, the SessionContext is used for accessing the information and container services.

## **19. Can remove () be a Stateless Session bean?**

Yes, remove () can be a Stateless Session bean because the life remains the same till the method is executed.

## **20. Is state maintained by a Stateless bean?**

A Stateless bean contains no-client specific state through client-invoked methods.

## **21. Can EJB made to handle multiple transactions?**

EJB can be made to handle multiple transactions by enabling multiple Entity beans to handle every database and one Session Bean to retain transaction with the Entity Bean.

## **22. Enlist the CallBack methods of Session Bean?**

```
1 public interface javax.ejb.SessionBean extends javax.ejb.EnterpriseBean {
2 Public abstract void ejbCreate();
3
4 public abstract void ejbRemove();
5
6 Public abstract void ejbActivate (); public abstract void setSessionContext(SessionContext ctx); public
7 abstract void ejbPassivate();
8
9 }
```

## **23. Enlist the CallBack methods of Entity Bean.**

```
1 public interface javax.ejb.EntityBean extends javax.ejb.EnterpriseBean {
2 public abstract void ejbRemove();
3
4 public abstract void ejbActivate();
5
6 public abstract void ejbStore();
7
8 public abstract void ejbPassivate(); public abstract void setEntityContext(EntityContext ctx); public abstract
9 void unsetEntityContext(); public abstract void ejbLoad();
10
11 }
```

## **24. How can one EJB be called from within another EJB?**

An EJB can be called within another EJB by using JNDI which can be used for locating the Home Interface and acquiring the instance.

## **25. Differentiate Conversational from Non-conversational interactions?**

The interaction between the client and the bean is called conversational while where multi method conversations are not held with clients it is known as non-conversational interactions.

## **26. Define ejb Create() and EjbPostCreate ()?**

When the method is called before the persistence storage is written with the bean state, it is ejbCreate ().

When the method is called after the persistence storage is written with the bean state, it is ejbPostCreate ().

## **27. Define EAR, WAR and JAR ?**

JAR files contain all EJB classes.

WAR files contain all servlets, web component pages, gif, html, beans, applets, classes and classes.

EAR files contain both JAR and WAR files.

## 28. Differentiate Phantom from Un-repeatable?

When data that did not existed before is inserted, it is read as phantom whereas when data that already existed is changed, un-repeatable occurs.

## 29. Define ACID Properties?

ACID is Atomicity, Consistency, Isolation and Durability.

Set of statements executed as a single unit of work

- Atomicity: Operations that are bundled together and projected a single unit of job.  
*Txn exists in a consistent state before and after the execution*
- Consistency: Guarantees that after a transaction has taken place, there will be consistency.  
*One txn executes independent of the other txn*
- Isolation: Helps protect viewing of other simultaneous incomplete transaction results.
- Durability: Ensures durability by keeping a transitional log by which the permanent data be recreated by again applying the steps involved. *Once COMMIT is fired txn survives even network failures*

## 30. What do you mean by ‘Hot deployment’?

The act of redeployment, deployment and un-deployment in Web logic when the server is running in EJB is called Hot Deployment.

## 31. How can a session bean be configured for transactions of bean-managed?

It can be done by setting transaction-attribute in the deployment sector or XML file.

## 32. Enlist the technologies embraced in J2EE.

The technologies embraced in J2EE are:

- Enterprise JavaBeansTM (EJBsTM)
- JavaServer PagesTM (JSPsTM)
- Java Servlets
- The Java Naming and Directory InterfaceTM (JNDITM)
- The Java Transaction API (JTA)
- CORBA
- The JDBC TM data access API.

## 33. What do you mean Enterprise JavaBeans (EJB) container?

Enterprise JavaBeans container helps in managing the implementation of enterprise beans applications of J2EE.

## 34. What do you mean by in-memory replication?

When the contents having the memory of a single physical m/c are simulated in all m/c in that cluster, that process is called memory replication.

### **35. Define Ripple Effect?**

During runtime, when the changes made in the various properties of server group, are propagated in every associated clone, this process is known as Ripple Effect.

### **36. Define Clone?**

Server group copies are defined as clone. But unlike Server Groups, clones are linked by means of nodes.

### **37. What do you mean by bean managed transaction?**

If the Container is not wanted by the developer for managing transactions, every database operation can be implemented to write the suitable JDBC code.

### **38. Differentiate, “find a method” from “select method” in EJB ?**

A persistent field is returned by the select method of an entity bean that is related. A remote or local interface is returned by the finder method.

### **39. What do you mean by abstract schema?**

An element of an entity's bean's deployment descriptor that defines the persistent fields of bean's and the relationship existing between them is known as Abstract Schema. It is specific for each entity beans which has managed persistence of container.

### **40. What do you mean by re-entrant? Can you say that session beans as re-entrant? Can entity beans be specified as re-entrant?**

If the entity bean is defined as re-entrant, then it is possible by multiple clients to associate with the Entity bean and get methods executed concurrently inside the entity bean. Synchronization is taken care of by container. There is an exception thrown when an entity beam is defined as non-re-entrant and numerous clients are connected to it concurrently to carry out a method.

### **41. What do you mean by EJB architecture?**

A non-visual component involving a transaction-oriented, distributed enterprise application is called Enterprise beans. They are characteristically deployed in containers of EJB and run on servers of EJB.

The three enterprise bean types are:

- Session Beans: These enterprise beans are non-persistent and can be stateless or stateful. If a conversational state is needed then, Stateful mode is preferred while Stateless paradigm is preferred for a single business process.
- Entity Beans: Entity Beans were presented in the earlier versions of EJB consisting of persistent data in distributed objects. They had the ability to be saved in different persistent data stores.
- Message Driven Beans: Asynchronous execution by means of messaging paradigm is supported. Follow the process of receiving and processing data. They are accessed only through messages and do not have a conversational state maintained.

#### **42. Write the basic requirement of a CMP entity based class in 2.0 from EJB 1.1?**

The basic requirement of a CMP is an abstract class which the container extends and gets the methods implemented required for managing the relationships.

#### **43. How can Enterprise JavaBeans be accessed from Active Server Pages?**

Enterprise JavaBeans can be accessed from Active Server Pages by:

- ‘Java 2 Platform’
- Enterprise Edition Client Access Services (J2EETM CAS) COM Bridge 1.0 which has been currently downloaded from the Sun Microsystems.

#### **44. Is having static initializer blocks legal in EJB?**

It is legal technically, but static initializer blocks have been used in executing pieces of code before the final execution of any method or constructor when a class is instantiated.

#### **45. What changes have been made in EJB 2.0 specifications?**

Changes that have been made in EJB 2.0 specification are:

- JMS is integrated with EJB.
- Message Driven Beans.
- Implementing additional Business methods.

#### **46. What do you mean by EJBDoclet?**

JavaDoc doclet, an open source is a doclet which generates good stuff related to EJB from comment tags of custom JavaDoc, which are embedded in the source file of EJB.

#### **47. What do you mean by EJB QL?**

A query language which provides navigation through a network comprising enterprise beans and objects which are dependent and are defined by methods of container managed persistence. EJB 2.0 was the platform for introduction of EJB QL. It defines methods of finder which are used for entity beans, which has container managed persistence and has portability across persistence managers and containers. It is helpful in two kinds of finder methods: Finder methods, which have Home interface and return objects of entity. Select methods, which remain unexposed for the client to see but which the Bean provider uses.

#### **48. How does EJB invocation take place?**

Home Object reference is retrieved from the Naming Service via JNDI. Home Object reference is returned to the client. The steps are:

- Created a new EJB Object via Home Object interface.
- Created an EJB Object from the Ejb Object.
- Returned an EJB Object reference to the client.
- Invoked business method by using EJB Object reference.
- Delegate requested to Bean (Enterprise Bean).

#### **49. Can more than a single table be mapped in CMP?**

No, more than one table cannot be mapped in a single CMP.

#### **50. Are entity beans allowed to create () methods?**

Yes, it is allowed in cases where data is not inserted by using Java application.

# Top 50 HTML Interview Questions

## 1) What is HTML?

HTML is short for HyperText Markup Language, and is the language of the World Wide Web. It is the standard text formatting language used for creating and displaying pages on the Web. HTML documents are made up of two things: the content and the tags that formats it for proper display on pages.

## 2) What are tags?

Content is placed in between HTML tags in order to properly format it. It makes use of the less than symbol (<) and the greater than symbol (>). A slash symbol is also used as a closing tag. For example:

```
1 sample
```

## 3) Do all HTML tags come in pair?

No, there are single HTML tags that does not need a closing tag. Examples are the <img> tag and <br> tags.

## 4) What are some of the common lists that can be used when designing a page?

You can insert any or a combination of the following list types:

- ordered list
- unordered list
- definition list
- menu list
- directory list

Each of this list types makes use of a different tag set to compose

## 5) How do you insert a comment in html?

Comments in html begins with “<!--” and ends with “-->”. For example:

```
1 <!-- A SAMPLE COMMENT -->
```

## 6) Do all character entities display properly on all systems?

No, there are some character entities that cannot be displayed when the operating system that the browser is running on does not support the characters. When that happens, these characters are displayed as boxes.

## 7) What is image map?

Image map lets you link to many different web pages using a single image. You can define shapes in images that you want to make part of an image mapping.

## 8) What is the advantage of collapsing white space?

White spaces are blank sequence of space characters, which is actually treated as a single space character in html. Because the browser collapses multiple space into a single space, you can indent lines of text without worrying about multiple spaces. This enables you to organize the html code into a much more readable format.

## 9) Can attribute values be set to anything or are there specific values that they accept?

Some attribute values can be set to only predefined values. Other attributes can accept any numerical value that represents the number of pixels for a size.

## 10) How do you insert a copyright symbol on a browser page?

<p>&copy; 2014 RapidTables.com</p>

To insert the copyright symbol, you need to type &copy; or &#169; in an HTML file. © 2014 RapidTables.com

## 11) How do you create links to sections within the same page?

Click [here](#chapter4) to read chapter 4.

Links can be created using the  tag, with referencing through the use of the number (#) symbol. For example, you can have one line as [BACK TO TOP](#topmost), which would result in the words “BACK TO TOP” appearing on the webpage and links to a bookmark named topmost. You then create a separate tag command like  somewhere on the top of the same webpage so that the user will be linked to that spot when he clicked on “BACK TO TOP”.

## 12) Is there any way to keep list elements straight in an html file?

Unordered or ordered list

By using indents, you can keep the list elements straight. If you indent each subnested list in further than the parent list that contains it, you can at a glance determine the various lists and the elements that it contains.

## 13) If you see a web address on a magazine, to which web page does it point?

Every web page on the web can have a separate web address. Most of these addresses are relative to the top-most web page. The published web address that appears within magazines typically points this top-most page. From this top level page, you can access all other pages within the web site.

## 14) What is the use of using alternative text in image mapping?



When you use image maps, it can easily become confusing and difficult to determine which hotspots corresponds with which links. Using alternative text lets you put a descriptive text on each hotspot link.

## 15) Do older html files work on newer browsers?

Yes, older html files are compliant to the HTML standard. Older files work on the newer browsers, though some features may not work.

## **16) Does a hyperlink apply to text only?**

No, hyperlinks can be used on text as well as images. That means you can convert an image into a link that will allow user to link to another page when clicked. Just surround the image within the `<a href="" ">...</a>` tag combinations.

## **17) If the user's operating system does not support the needed character, how can the symbol be represented?**

In cases wherein their operating system does not support a particular character, it is still possible to display that character by showing it as an image instead.

## **18) How do you change the number type in the middle of a list?**

The `<li>` tag includes two attributes – type and value. The type attribute can be used to change the numbering type for any list item. The value attribute can change the number index.

## **19) What are style sheets?**

Style sheets enable you to build consistent, transportable, and well-defined style templates. These templates can be linked to several different web pages, making it easy to maintain and change the look and feel of all the web pages within a site.

## **20) What bullet types are available?**

With ordered lists, you can select to use a number of different list types including alphabetical and Roman numerals. The type attribute for unordered lists can be set to disc, square, or circle.

## **21) How do you create multicolored text in a webpage?**

To create text with different colors, use the `<font color="color">...</font>` tags for every character that you want to apply a color. You can use this tag combination as many times as needed, surrounding a single character or an entire word.

## **22) Why are there both numerical and named character entity values?**

The numerical values are taken from the ASCII values for the various characters, but these can be difficult to remember. Because of this, named character entity values were created to make it easier for web page designers to use.

**23) Write a HTML table tag sequence that outputs the following:**

50 pcs 100 500

10 pcs 5 50

```
1 <table>
2 <tr>
3 <td>50 pcs</td>
4 <td>100</td>
5 <td>500</td>
6 </tr>
7 <tr>
8 <td>10 pcs</td>
9 <td>5</td>
10 <td>50</td>
11 </tr>
12 </table>
```

**24) What is the advantage of grouping several checkboxes together?**

Although checkboxes don't affect one another, grouping checkboxes together helps to organize them. Checkbox buttons can have their own name and do not need to belong to a group. A single web page can have many different groups of checkboxes.

**25) What will happen if you overlap sets of tags?**

If two sets of html tags are overlapped, only the first tag will be recognized. You will recognize this problem when the text does not display properly on the browser screen.

**26) What are applets?**

Applets are small programs that can be embedded within web pages to perform some specific functionality, such as computations, animations, and information processing. Applets are written using the Java language.

**27) what if there is no text between the tags or if a text was omitted by mistake? Will it affect the display of the html file?**

If there is no text between the tags, then there is nothing to format, so no formatting will appear. Some tags, especially tags without a closing tag like the `<img>` tag, do not require any text between them.

**28) Is it possible to set specific colors for table borders?**

You can specify a border color using style sheets, but the colors for a table that does not use style sheets will be the same as the text color.

## **29) How do you create a link that will connect to another web page when clicked?**

To create hyperlinks, or links that connect to another web page, use the href tag. The general format for this is:

`<a href="site">text</a>`

Replace “site” with the actual page url that is supposed to be linked to when the text is clicked.

## **30) What other ways can be used to align images and wrap text?**

Tables can be used to position text and images. Another useful way to wrap text around an image is to **use style sheets**.

## **31) Can a single text link point to two different web pages?**

**No.** The <a> tag can accept only a single href attribute, and it can point to only a single web page.

## **32) What is the difference between the directory and menu lists and the unordered list?**

The key differences is that the directory and menu lists **do not include attributes for changing the bullet style**.

## **33) Can you change the color of bullets?**

The bullet color is always the same as that of the first character in the list item. If you **surround the <li> and the first character with a set of <font> tags** with the color attribute set, the bullet color and the first character will be a different color from the text.

## **34) What are the limits of the text field size?**

The default size for a text field is **around 13 characters**, but if you include the size attribute, you can set the size value to be **as low as 1**. The **maximum size value will be determined by the browser width**. If the **size attribute is set to 0**, the size will be **set to the default size of 13 characters**.

## **35) Do <th> tags always need to come at the start of a row or column?**

**Any <tr> tag can be changed to a <th> tag.** This causes the text contained within the <th> tag to be displayed as bold in the browser. Although **<th> tags are mainly used for headings**, they do not need to be used exclusively for headings.

## **36) What is the relationship between the border and rule attributes?**

**Default cell borders, with a thickness of 1 pixel, are automatically added between cells** if the **border attribute is set to a nonzero value**. Likewise, If the border attribute is not included, a default 1-pixel border appears if the rules attribute is added to the <table> tag.

### **37) What is a marquee?**

A marquee allows you to put a scrolling text in a web page. To do this, place whatever text you want to appear scrolling within the <marquee> and </marquee> tags.

### **38) How do you create a text on a webpage that will allow you to send an email when clicked?**

To change a text into a clickable link to send email, use the mailto command within the href tag. The format is as follows:

1 <A HREF="mailto:youremailaddress">text to be clicked</A>

### **39) Are <br> tags the only way to separate sections of text?**

No, the <br> tag is only one way to separate lines of text. Other tags, like the <p> tag and <blockquote> tag, also separate sections of text.

### **40) Are there instances where text will appear outside of the browser?**

By default, the text is wrapped to appear within the browser window. However, if the text is part of a table cell with a defined width, the text could extend beyond the browser window.

### **41) How are active links different from normal links?**

The default color for normal and active links is blue. Some browsers recognize an active link when the mouse cursor is placed over that link; others recognize active links when the link has the focus. Those that don't have a mouse cursor over that link is considered a normal link.

### **42) Do style sheets limit the number of new style definitions that can be included within the brackets?**

Style sheets do not limit the number of style definitions that can be included within the brackets for a given selector. Every new style definition, however, must be separated from the others by a semicolon symbol.

### **43) Can I specify fractional weight values such as 670 or 973 for font weight?**

Implementation largely depends on the browser, but the standard does not support fractional weight values. Acceptable values must end with two zeroes.

### **44) What is the hierarchy that is being followed when it comes to style sheets?**

If a single selector includes three different style definitions, the definition that is closest to the actual tag takes precedence. Inline style takes priority over embedded style sheets, which takes priority over external style sheets.

#### **45) Can several selectors with class names be grouped together?**

You can define several selectors with the same style definition by separating them with commas. This same technique also works for selectors with class names.

#### **46) What happens if you open the external CSS file in a browser?**

If you try to open the external CSS file in a browser, the browser cannot open the file, because the file has a different extension. The only way to use an external CSS file is to reference it using <link> tag within another html document.

#### **47) How do you make a picture into a background image of a web page?**

To do this, place a tag code after the </head> tag as follows:

```
1 <body background = "image.gif">
```

Replace image.gif with the name of your image file. This will take the picture and make it the background image of your web page.

#### **48) What happens if the list-style-type property is used on a non-list element like a paragraph?**

If the list-style-type property is used on a non-list element like a paragraph, the property will be ignored and have no effect on the paragraph.

#### **49) When is it appropriate to use frames?**

Frames can make navigating a site much easier. If the main links to the site are located in a frame that appears at the top or along the edge of the browser, the content for those links can be displayed in the remainder of the browser window.

#### **50) What happens if the number of values in the rows or cols attribute doesn't add up to 100 percent?**

The browser sizes the frames relative to the total sum of the values. If the cols attribute is set to 100%, 200%, the browser displays two vertical frames with the second being twice as big as the first.

An average recruiter has 6 seconds to check your resume... will you be noticed?

[Check Premium Resumes](#)



## Top 50 J2EE interview questions

### 1) What is J2EE?

J2EE means Java 2 Enterprise Edition. The functionality of J2EE is developing multitier web-based applications .The J2EE platform is consists of a set of services, application programming interfaces (APIs), and protocols.

### 2) What are the four components of J2EE application?

- Application clients components.
- Servlet and JSP technology are web components.
- Business components (JavaBeans).
- Resource adapter components

### 3) What are types of J2EE clients?

- Applets
- Application clients
- Java Web Start-enabled clients, by Java Web Start technology.
- Wireless clients, based on MIDP technology.

### 4) What are considered as a web component?

Java Servlet and Java Server Pages technology components are web components. Servlets are Java programming language that dynamically receive requests and make responses. JSP pages execute as servlets but allow a more natural approach to creating static content.

### 5) What is JSF?

JavaServer Faces (JSF) is a user interface (UI) designing framework for Java web applications. JSF provide a set of reusable UI components, standard for web applications.JSF is based on MVC design pattern. It automatically saves the form data to server and populates the form date when display at client side.

### 6) Define Hash table

---

HashTable is just like Hash Map,Collection having key(Unique),value pairs. Hashtable is a collection Synchronized object .It does not allow duplicate values but it allows null values.

## 7) What is Hibernate?

Hibernate is a open source object-relational mapping and query service. In hibernate we can write HQL instead of SQL which save developers to spend more time on writing the native SQL. Hibernate has more powerful association, inheritance, polymorphism, composition, and collections. It is a beautiful approach for persisting into database using the java objects. Hibernate also allows you to express queries using java-based criteria .

## 8 ) What is the limitation of hibernate?

- Slower in executing the queries than queries are used directly.
- Only query language support for composite keys.
- No shared references to value types.

## 9) What are the advantage of hibernate.

- Hibernate is portable i mean database independent, Vendor independence.
- Standard ORM also supports JPA
- Mapping of Domain object to relational database.
- Hibernate is better then plain JDBC.
- JPA provider in JPA based applications.

## 10) What is ORM?



ORM stands for Object-Relational mapping. The objects in a Java class which is mapped in to the tables of a relational database using the metadata that describes the mapping between the objects and the database. It works by transforming the data from one representation to another.

## 11) Difference between save and saveorupdate

**save()** - This method in hibernate is used to stores an object into the database. It insert an entry if the record doesn't exist, otherwise not.

**saveorupdate ()** -This method in the hibernate is used for updating the object using identifier. If the identifier is missing this method calls save(). If the identifier exists, it will call update method.

**12) Difference between load and get method?**

**load()** can't find the object from cache or database, an exception is thrown and the **load()** method never returns null.

**get()** method returns null if the object can't be found. The **load()** method may return a proxy instead of a real persistent instance **get()** never returns a proxy.

**13) How to invoke stored procedure in hibernate?**

```
{ ? = call thisISTheProcedure() }
```

**14) What are the benefits of ORM?**

- Productivity
- Maintainability
- Performance
- Vendor independence

**15) What the Core interfaces are of hibernate framework?**

- Session Interface
- SessionFactory Interface
- Configuration Interface
- Transaction Interface
- Query and Criteria Interface

**16) What is the file extension used for hibernate mapping file?**

The name of the file should be like this : filename.hbm.xml

**17) What is the file name of hibernate configuration file?**

The name of the file should be like this : hibernate.cfg.xml

**18) How hibernate is database independent explain?**

Only changing the property  
[crayon-54f5ae87a2662895366223/]

full database can be replaced.

**19) How to add hibernate mapping file in hibernate configuration file?**

---

By

## 20) Define connection pooling?

Connection pooling is a mechanism reuse the connection which contains the number of already created object connection. So whenever there is a necessary for object, this mechanism is used to directly get objects without creating it.

## 21) What is the hibernate proxy?

An object proxy is just a way to avoid retrieving an object until you need it. Hibernate 2 does not proxy objects by default.

## 22) What do you create a SessionFactory?

[crayon-54f5ae87a266a889763982/]

## 23) What is HQL?

HQL stands for Hibernate Query Language. Hibernate allows to the user to express queries in its own portable SQL extension and this is called as HQL. It also allows the user to express in native SQL.

## 24) What are the Collection types in Hibernate ?

Set, List, Array, Map, Bag

## 25) What is a thin client?

A thin client is a program interface to the application that does not have any operations like query of databases, execute complex business rules, or connect to legacy applications.

## 26) Differentiate between .ear, .jar and .war files.

**.jar files:** These files are with the .jar extension. The .jar files contains the libraries, resources and accessories files like property files.

**.war files:** These files are with the .war extension. The .war file contains jsp, html, javascript and other files for necessary for the development of web applications.

**.ear files:** The .ear file contains the EJB modules of the application.

## 27) What are the JSP tag?

In JSP tags can be divided into 4 different types.

- Directives
- Declarations

- **Scriptlets**
- **Expressions**

## 28) How to access web.xml init parameters from jsp page?

For example, if you have:

**Id this is the value**

**You can access this parameter**

**Id:**

## 29) What are JSP Directives?

- **1.page Directives**
- **2. include Directives:**
- **3. taglib Directives**

## 30) What is the EAR file?

An EAR file is a JAR file with an .ear extension. A J2EE application with all of its modules is delivered in EAR file.

## 31) What will happen when you compile and run the following code?

[crayon-54f5ae87a266e979670789/]

## 32) What is Struts?

Struts framework is a Model-View-Controller(MVC) **architecture for designing large scale applications**. Which is combines of Java Servlets, JSP, Custom tags, and message. Struts helps you to create an extensible development environment for your application, based on published standards and proven design patterns. Model in many **applications represent the internal state of the system as a set of one or more JavaBeans**. The **View** is most often constructed using **JavaServer Pages (JSP) technology**. The **Controller** is focused on **receiving requests from the client and producing the next phase of the user interface to an appropriate View component**. The primary component of the Controller in the framework is a  **servlet of class ActionServlet**. This servlet is configured by defining a **set of ActionMappings**.

## 33.What is ActionErrors?

ActionErrors object that encapsulates any validation errors that have been found. If no errors are found, return null or an ActionErrors object with no recorded error messages. The default implementation attempts to forward to the HTTP version of this method. Holding request parameters mapping and request and returns set of validation errors, if validation failed; an empty set or null

## 34) What is ActionForm?

ActionForm is a Java bean that associates one or more ActionMappings. A java bean become

---

FormBean when extend org.apache.struts.action.ActionForm class. ActionForm object is automatically populated on the server side which data has been entered by the client from UI. ActionForm maintains the session state for web application.

### 35) What is action mapping??

In action mapping we specify action class for particular url ie path and different target view ie forwards on to which request response will be forwarded. The **ActionMapping** represents the information that the **ActionServlet** knows about the mapping of a particular request to an instance of a particular **Action** class. The **mapping** is passed to the **execute()** method of the **Action** class, enabling access to this information directly.

### 36) What is the MVC on struts.

**MVC** stands **Model-View-Controller**.

**Model:** Model in many applications represent the internal state of the system as a set of one or more JavaBeans.

**View:** The View is most often constructed using JavaServer Pages (JSP) technology.

**Controller:** The Controller is focused on receiving requests from the client and producing the next phase of the user interface to an appropriate View component. The primary component of the Controller in the framework is a servlet of class ActionServlet. This servlet is configured by defining a set of ActionMappings.

### 37) What are different modules in spring?

There are seven core modules in spring

- The Core container module
- O/R mapping module (Object/Relational)
- DAO module
- Application context module
- Aspect Oriented Programming
- Web module
- MVC module

### 38) What is Bean Factory, have you used XMLBean factory?

[crayon-54f5ae87a2673481566042/]

### 39) What is Spring?

Spring is a light weight open source framework for the development of enterprise application that resolves the complexity of enterprise application development also providing a cohesive

---

framework for J2EE application development. Which is primarily based on IOC (inversion of control) or DI (dependency injection) design pattern.

#### **40) Functionality of ActionServlet and RequestProcessor?**

- Receiving the HttpServletRequest
- Populating JavaBean from the request parameters
- Displaying response on the web page Issues
- Content type issues handling
- Provide extension points

#### **41) ActionServlet, RequestProcessor and Action classes are the components of Controller**

#### **42) What is default scope in Spring?**

Singleton.

#### **43) What are advantages of Spring usage?**

- Pojo based programming enables reuse component.
- Improve productivity and subsequently reduce development cost.
- Dependency Injection can be used to improve testability.
- Spring required enterprise services without a need of expensive application server.
- It reduces coupling in code and improves maintainability.

#### **44)What are the Benefits Spring Framework ?**

- Light weight container
- Spring can effectively organize your middle tier objects
- Initialization of properties is easy ? no need to read from properties file
- application code is much easier to unit test
- Objects are created Lazily , Singleton - configuration
- Spring's configuration management services can be used in any architectural layer, in whatever runtime environment

#### **45) Lifecycle interfaces in spring ?**

[crayon-54f5ae87a2676426540239/]

#### **46) How to Create Object without using the keyword "new" in java?**

Without new the Factory methods are used to create objects for a class. For example  
Calender c=Calender.getInstance();

here Calender is a class and the method getInstance() is a Factory method which can create object for Calender class.

**47) What is servlet?**

Servlets is a server side components that provide a powerful mechanism for developing server side programs. Servlets is a server as well as platform-independent and Servlets are designed for a various protocols. Most commonly used HTTP protocols. Servlets uses the classes in the java packages javax.servlet, javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse, javax.servlet.http.HttpSession;. All servlets must implement the Servlet interface, which defines life-cycle methods.

**48) Servlet is pure java object or not?**

Yes, pure java object.

**49) What are the phases of the servlet life cycle?**

The life cycle of a servlet consists of the following phases:

- Servlet class loading
- Servlet instantiation
- the init method
- Request handling (call the service method)
- Removal from service (call the destroy method)

**50) What must be implemented by all Servlets?**

The Servlet Interface must be implemented by all servlets

Guru99 provides [FREE ONLINE TUTORIAL](#) on Various courses like

<a href="#">PHP</a>	<a href="#">Java</a>	<a href="#">Linux</a>	<a href="#">Apache</a>
<a href="#">Perl</a>	<a href="#">SQL</a>	<a href="#">VB Script</a>	<a href="#">JavaScript</a>
<a href="#">Accounting</a>	<a href="#">Ethical Hacking</a>	<a href="#">Cloud Computing</a>	<a href="#">Jmeter</a>
<a href="#">Manual Testing</a>	<a href="#">QTP</a>	<a href="#">Selenium</a>	<a href="#">Test Management</a>
<a href="#">Load Runner</a>	<a href="#">Quality Center</a>	<a href="#">Mobile Testing</a>	<a href="#">Live Selenium</a>

[Project](#)

[Enterprise Testing](#)

[Live Testing Project](#)

[Sap & All Modules](#)



---

Copyrighted Material

# TOP 50 JSP

## 1. Explain JSP and tell its uses.

JSP stands for Java Server Pages. It is a presentation layer technology independent of platform. It comes with SUN's J2EE platforms. They are like HTML pages but with Java code pieces embedded in them. They are saved with a .jsp extension. They are compiled using JSP compiler in the background and generate a Servlet from the page.

## 2. What is the requirement of a tag library?

A collection of custom tags is called a Tag Library. Recurring tasks are handled more easily and reused across multiple applications to increase productivity. They are used by Web Application designers who focus on presentation rather than accessing database or other services. Some popular libraries are String tag library and Apache display tag library.

## 3. Explain JSP Technology.

JSP is a standard extension of Java and is defined on top of Servlet extensions. Its goal is to simplify management and creation of dynamic web pages. It is platform-independent, secure, and it makes use of Java as a server side scripting language.

## 4. Explain Implicit objects in JSP.

Objects created by web container and contain information regarding a particular request, application or page are called Implicit Objects. They are :

1)response

2)exception

3)application

4)request

5)session

6)page

7)out

8)config

## 9)pageContext

### 5. How can multiple submits due to refresh button clicks be prevented?

Using a Post/Redirect/Get or a PRG pattern, this problem can be solved.

1. A form filled by the user is submitted to the server using POST or GET method. The state in the database and business model are updated.
2. A redirect response is used to reply by the servlet for a view page.
3. A view is loaded by the browser using the GET command and no user data is sent. This is safe from multiple submits as it is a separate JSP page.

### 6. Is JSP technology extensible?

Yes, JSP is easily extensible by use and modification of tags, or custom actions, encapsulated in tag libraries.

### 7. Differentiate between response.sendRedirect(url) and <jsp:forward page = ...> .

<jsp:forward> element forwards the request object from 1 JSP file to another. Target file can be HTML, servlet or another JSP file, but it should be in the same application context as forwarding JSP file.

sendRedirect send HTTP temporary redirect response to the browser. The browser then creates a new request for the redirected page. It kills the session variables.

### 8. Can a subsequent request be accessed with one's servlet code, if a request attribute is already sent in his JSP?

The request goes out of scope, thus, it cannot be accessed. However, if a request attribute is set in one's servlet, then it can be accessed in his JSP.

A JSP is a server side component and the page is translated to a Java servlet, and then executed. Only HTML code is given as output.

### 9. How to include static files in a JSP page?

Static pages are always included using JSP include directive. This way the inclusion is performed in the translation phase once. Note that a relative URL must be supplied for file attribute. Although static resources may be included, it is not preferred as each request requires inclusion.

### 10. Why is it that JComponent have add() and remove() methods but Component doesn't?

JComponent is a **subclass of Container**. It contains other Components and JComponents.

## 11. How can a thread safe JSP page be implemented?

It can be done by having them implemented by the **SingleThreadModel Interface**. Add `<%@page isThreadSafe="false" %>` directive in the JSP page.

## 12. How can the output of JSP or servlet page be prevented from being cached by the browser?

Using appropriate HTTP header attributes to prevent the dynamic content output by a JSP page from being cached by the browser.

## 13. How to restrict page errors display in a JSP page?

By setting up an “ErrorPage” attribute of PAGE directory to the name of the error page in the JSP page, and then in the error jsp page set “isErrorpage=”TRUE”, Errors can be stopped from getting displayed.

## 14. What are JSP Actions?

They are XML tags, **which direct the server to using existing components** or control behavior of JSP Engine. They consist of a typical prefix of “jsp:” and action name.

- 1 `<jsp:include/>`
- 2 `<jsp:getProperty/>`
- 3 `<jsp:forward/>`
- 4 `<jsp:setProperty/>`
- 5 `<jsp:usebean/>`
- 6 `<jsp:plugin/>`

## 15. Differentiate between `<jsp:include page=...>` and `<%@include file=...>`.

Both these tags include information from 1 page to another.

The first tag acts as a function call between two Jsp’s. It is **executed each time client page is accessed by the client**. It **is useful to modularize the web application**. New content is included in the output.

The second tag content of file is textually embedded having similar directive. **The changed content is not included in the output**. It is helpful when code from one jsp is required by several jsp’s.

## 16. Can constructor be used instead of `init()`, to initialize servlet?

Yes, it is possible. But it is not preferred because `init()` was developed because earlier Java versions could not invoke constructors with arguments dynamically. So they could not assign a `servletConfig`. Today, however, servlet containers still call only no-arg constructor. So there is no access to `servletContext` or `servletConfig`.

## 17. Explain lifecycle methods.

1) **jsplnit()**: The container calls this to initialize servlet instance. It is called only once for the servlet instance and preceded every other method.

2) **\_jspService()**: The container calls this for each request and passes it on to the objects.

3) **jspDestroy()**: It is called by the container just before destruction of the instance.

## 18. Explain JSP Output comments?

They are comments that can be viewed in HTML Source File.

## 19. Define Expression

Expression tag is used to insert Java values directly in the output. Its syntax is <%=expression%>

It contains a scripting language expression that is evaluated, then converted to a string, and then inserted where the expression comes in JSP file. Only when there is a IS-A relation, use Inheritance. If you want just code reuse, use Composition. Use Interfaces if you want polymorphism.

**20. Define Composition.** Contained object is not dependent on container object, when container is destroyed contained remains undestroyed. E.g. School-students

Composition has a stronger relationship with the object than Aggregation.

**Composition:** Contained object is dependent on container object, when container is destroyed contained is also destroyed. E.g. House-rooms Has-A relationship

## 21. Define JSP Scriptlet.

It a JSP tag that encloses Java code in JSP pages. Their syntax is <% %>. Code written in scriptlet executes every time the program is run.

## 22. How can information from one JSP be passed to another JSP?

The tag <Jsp:param> allows us to pass information between multiple Jsp's.

## 23. Explain the uses of <jsp:usebean> tag.

```
1 <jsp:useBean>
2
3 id="beanInstName"
4
5 scope= "page | application"
6
7 class="ABC.class" type="ABC.class"
8
9 </jsp:useBean>
```

This tag creates an instance of java bean. It firstly tries to find if bean instance already exist and assign stores a reference in the variable. Type is also specified; otherwise it instantiates from the specified class storing a reference in the new variable.

## 24. Explain handling of runtime exceptions.

Errorpage attribute is used to uncatch the run-time exceptions forwarded automatically to an error processing page.

It redirects the browser to JSP page error.jsp if any uncaught exception is faces during request handling. It is an error processing page.

## 25. Why does `_jspService()` start with an ‘\_’ but other lifecycle methods do not?

Whatever content made in a jsp page goes inside the `_jspService()` method by the container. If it is override, the compiler gives an error, but the other 2 lifecycles can be easily override. So ‘\_’ shows that we cannot override this method.

## 26. Explain the various scope values for `<jsp:useBean>` tag.

`<jsp:useBean>` tag is used to use any java object in the jsp page. Some scope values are :

1) application

2) request

3) page

4) session

## 27. Show the 2 types of comments in JSP.

The 2 types are :

```
1 <%--JSP Comment--%>
2 <!--HTML comment-->
```

## 28. Can Static method be Override?

We can declare static methods with same signature in subclass, but it is not considered overriding as there won't be any run-time polymorphism. Hence the answer is ‘No’.

## 29. Explain JSP directives.

JSP directives are messages to JSP Engine. They serve as a message from page to container and control the processing of the entire page. They can set global values like class declaration. They do not produce output and are enclosed in `<%@....%>`

## 30. Explain pge Directives.

Page Directives inform the JSP Engine about headers and facilities that the page receives from the environment. It is found at the top of all JSP pages. Its syntax is <%@ page attribute="value">

### 31. Show attributes of page directives.

- 1)Session : It shows if a session data is available to the page.
- 2)Import : it shows packages that are imported.
- 3)isELIgnored : It shows whether EL expressions are ignored when JSP translates into a servlet.
- 4)contentType : it allows the user to specify the content type of page.

### 32. What is Include directive?

The include directive statically inserts the contents of a resource into the current JSP. It helps in the reuse of code without duplication. and includes contents of the file at translation time. Its syntax is as follows <%@ include file="Filename"%>.

### 33. What are standard actions in JSP?

They affect overall runtime behaviour of a page and response sent to the client. They are used to include a file at request time, to instantiate a JavaBean or find one. They are also used to generate a browser-specific code or forward a request to a new page.

### 34. Explain the jsp:setProperty action.

It is used to give values to properties of beans that have been referenced beforehand.

```
<jsp:useBean id="ABC" .../>
```

..

```
<jsp:setProperty name="ABC" property="myProperty" ...>
```

jsp:setproperty is executed even if a new bean is instantiated or existing bean is found.

By adding </jsp.useBean> at the end of the code, the condition for execution is inverted i.e. It is not executed if existing object was found and only if a new object was instantiated.

### 35. Define Static Block.

It is used to start the static data member. It is executed before classloading.

### **36. Explain jsp:plugin action.**

This action helps in insertion of a specific object in the browser or embed the element needed to specify the running of applet using Java plugin.

### **37. Explain client and server side validation.**

Javascript is used for the client-side validation. It takes place within the browser. Javascript is used to submit the form data if validation is successful. Validation errors require no extra network trip because form cannot be submitted.

Validation is also carried out in the server after submission. If validation fails, extra network trip is required to resend the form to the client.

### **38. What is Translation Phase?**

JSP engine translates and compiles a JSP file to a servlet. This servlet moves to the execution phase where requests and responses are handled. They are compiled for the first time they are accessed unless manually compiled ahead of time. The manual or explicit compilation is useful for long and convoluted programs.

### **39. Perform a Browser Redirection from a JSP Page.**

```
<% response.sendRedirect(URL); %>
```

or we can change the location of the HTTP header attribute as follows:

```
<% response.setStatus(HttpServletRequest.SC_MOVED_PERMANENTLY); response.setHeader(URL); %>
```

### **40. Give uses of Object Cloning.**

Object cloning is used to create an exact copy of an object by typing the same code or using various other techniques.

### **41. How to forward a request to another source.**

```
1 <jsp:forward page="/Page2.jsp" />
```

### **42. How can Automatic creation of session be prevented in a JSP page?**

JSP page automatically create sessions for requests. By typing the following, it can be avoided.

```
1 <%@ page session="false" %>
```

### **43. How can you avoid scriptlet code in JSP?**

JavaBeans or Custom Tags can be used instead of scriptlet code.

#### 44. Explain the `jspDestroy()` method.

Whenever a JSP page is about to be destroyed, the container invokes the `jspDestroy()` method from the `javax.servlet.jsp.JspPage` interface. Servlets `destroy` methods are similar to it. It can be easily overridden to perform cleanup, like when closing a database connection.

#### 45. Explain the `<jsp:param>` action.

It is an action used with include or forward standard actions. It helps in passing the parameter names and values to a resource.

#### 46. Explain static method.

A static method is of the class and not the object of a class. It can be invoked without instance of a class. Static members can also access the static data and change its value.

#### 47. How to disable scripting?

Scripting can be easily disabled by setting `scripting-invalid` element of the deployment descriptor to true. It is a sub-element of property group. Its can be false as well.

#### 48. Define JSP Declaration.

JSP Declaration are tags used in declaring variables. They are enclosed in `<% ! %>` tag. They are used in declaring functions and variables.

```
1 <%@page contentType="text/html" %>
2 <html>
3 <body>
4 <%!
5 int a=0;
6 private int getCount(){
7 a++;
8 return a;
9 }
10 <p>Values of a are:</p>
11 <p><%=getCount()%></p>
12 </body>
```

```
25
26 </html>
27
```

#### 49. How can HTML Output be prevented from being cached?

```
1 <%
2
3 response.setHeader("Cache-Control", "no-store");
4
5 response.setDateHeader("Expires", 0);
6
7 %>
```

#### 50. How is JSP better than Servlet technology?

JSP is a **technology on the server's side to make content generation simple**. They are document centric, whereas  **servlets are programs**. A Java server page can contain fragments of Java program, which execute and instantiate Java classes. However, they occur inside HTML template file. It provides the framework for development of a Web Application.

**Learn Java/J2EE core concepts and design/coding issues**

With

# **Java/J2EE Job Interview Companion**

By

**K.Arulkumaran**

## **Technical Reviewers**

Craig Malone  
Lara D'Albreo  
Stuart Watson

## **Acknowledgements**

A. Sivayini  
R.Kumaraswamipillai

## **Cover Design**

K. Arulkumaran  
A.Sivayini

**Java/J2EE  
Job Interview Companion**

Copy Right 2005 K.Arulkumaran

The author has made every effort in the preparation of this book to ensure the accuracy of the information. However, information in this book is sold without warranty either express or implied. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by this book.

## Outline

SECTION	DESCRIPTION
	<p><b>What this book will do for you?</b></p> <p><b>Motivation for this book</b></p> <p><b>Key Areas index</b></p>
<b>SECTION 1</b>	<p>Interview questions and answers on:</p> <p><b>Java</b></p> <ul style="list-style-type: none"> <li>▪ Language Fundamentals</li> <li>▪ Swing</li> <li>▪ Applet</li> <li>▪ Performance and memory Leaks.</li> <li>▪ Personal</li> </ul> <p><b>Object-Oriented Analysis (OOA)</b> is the procedure of identifying software engineering requirements and developing software specifications in terms of a software system's object model, which comprises of interacting objects.</p> <p>The <b>benefits of using the object model</b> are:</p> <ul style="list-style-type: none"> <li>* It helps in faster development of software.</li> <li>* It is easy to maintain.</li> </ul>
<b>SECTION 2</b>	<p>Interview questions and answers on:</p> <p><b>Enterprise Java</b></p> <ul style="list-style-type: none"> <li>▪ J2EE</li> <li>▪ Servlet</li> <li>▪ JSP</li> <li>▪ JDBC</li> <li>▪ JNDI</li> <li>▪ RMI</li> <li>▪ EJB</li> <li>▪ JMS</li> <li>▪ XML</li> <li>▪ SQL, Database tuning and O/R mapping</li> <li>▪ RUP &amp; UML</li> <li>▪ Struts</li> <li>▪ Web and Application servers.</li> <li>▪ Best practices and performance considerations.</li> <li>▪ Testing and deployment.</li> <li>▪ Personal</li> </ul> <p><b>The Unified Modeling Language (UML)</b> is a graphical language for OOAD that gives a standard way to write a software system's blueprint. It helps to visualize, specify, construct, and document the artifacts of an object-oriented system. It is used to depict the structures and the relationships in a complex system.</p> <p>The <b>three building blocks of UML</b> are:</p> <p style="text-align: center;">Things Relationships Diagrams</p>
<b>SECTION 3</b>	<p>Putting it all together section.</p> <p><b>How would you go about...?</b></p> <ol style="list-style-type: none"> <li>1. How would you go about documenting your Java/J2EE application?</li> <li>2. How would you go about designing a Java/J2EE application?</li> <li>3. How would you go about identifying performance problems and/or memory leaks in your Java application?</li> <li>4. How would you go about minimising memory leaks in your Java/J2EE application?</li> <li>5. How would you go about improving performance of your Java/J2EE application?</li> <li>6. How would you go about identifying any potential thread-safety issues in your Java/J2EE application?</li> <li>7. How would you go about identifying any potential transactional issues in your Java/J2EE application?</li> <li>8. How would you go about applying the Object Oriented (OO) design concepts in your Java/J2EE</li> </ol>

	<p>application?</p> <ol style="list-style-type: none"> <li>9. How would you go about applying the UML diagrams in your Java/J2EE project?</li> <li>10. How would you go about describing the software development processes you are familiar with?</li> <li>11. How would you go about applying the design patterns in your Java/J2EE application?</li> <li>12. How would you go about determining the enterprise security requirements for your Java/J2EE application?</li> <li>13. How would you go about describing the open source projects like JUnit (unit testing), Ant (build tool), CVS (version control system) and log4J (logging tool) which are integral part of most Java/J2EE projects?</li> <li>14. How would you go about describing Web services?</li> </ol>
<b>SECTION 4</b>	<p><b>Emerging Technologies/Frameworks</b></p> <ul style="list-style-type: none"> <li>▪ Test Driven Development (<b>TDD</b>).</li> <li>▪ Aspect Oriented Programming (<b>AOP</b>).</li> <li>▪ Inversion of Control (<b>IOC</b>) (Also known as <b>Dependency Injection</b>).</li> <li>▪ Annotations or attributes based programming (xdoclet etc).</li> <li>▪ Spring framework.</li> <li>▪ Hibernate framework.</li> <li>▪ EJB 3.0.</li> <li>▪ JavaServer Faces (<b>JSF</b>) framework.</li> </ul>
<b>SECTION 5</b>	<p><b>Sample interview questions ...</b></p> <ul style="list-style-type: none"> <li>▪ <b>Java</b></li> <li>▪ <b>Web Components</b></li> <li>▪ <b>Enterprise</b></li> <li>▪ <b>Design</b></li> <li>▪ <b>General</b></li> </ul>
	<b>GLOSSARY OF TERMS</b>
	<b>RESOURCES</b>
	<b>INDEX</b>

## Table of contents

<b>Outline</b>	<b>3</b>
<b>Table of contents</b>	<b>5</b>
<b>What this book will do for you?</b>	<b>7</b>
<b>Motivation for this book</b>	<b>8</b>
<b>Key Areas Index</b>	<b>10</b>
<b>Java – Interview questions &amp; answers</b>	<b>11</b>
Java – Language Fundamentals	12
Java – Swing	44
Java – Applet	48
Java – Performance and Memory leaks	50
Java – Personal	53
Java – Key Points	56
<b>Enterprise Java – Interview questions &amp; answers</b>	<b>59</b>
Enterprise - J2EE	60
Enterprise - Servlet	69
Enterprise - JSP	77
Enterprise - JDBC	83
Enterprise – JNDI & LDAP	87
Enterprise - RMI	90
Enterprise – EJB 2.x	94
Enterprise - JMS	110
Enterprise - XML	114
Enterprise – SQL, Tuning and O/R mapping	119
Enterprise - RUP & UML	126
Enterprise - Struts	133
Enterprise - Web and Application servers	137
Enterprise - Best practices and performance considerations	139
Enterprise – Logging, testing and deployment	141
Enterprise - Personal	144
Enterprise – Software development process	144
Enterprise – Key Points	146
<b>How would you go about...?</b>	<b>151</b>
Q 01: How would you go about documenting your Java/J2EE application?	152
Q 02: How would you go about designing a Java/J2EE application?	153
Q 03: How would you go about identifying performance and/or memory issues in your Java/J2EE application?	156
Q 04: How would you go about minimising memory leaks in your Java/J2EE application?	157
Q 05: How would you go about improving performance in your Java/J2EE application?	157
Q 06: How would you go about identifying any potential thread-safety issues in your Java/J2EE application?	158
Q 07: How would you go about identifying any potential transactional issues in your Java/J2EE application?	159
Q 08: How would you go about applying the Object Oriented (OO) design concepts in your Java/J2EE application?	160
Q 09: How would you go about applying the UML diagrams in your Java/J2EE project?	162

Q 10: How would you go about describing the software development processes you are familiar with? _____	163
Q 11: How would you go about applying the design patterns in your Java/J2EE application? _____	165
Q 12: How would you go about determining the enterprise security requirements for your Java/J2EE application? _____	194
Q 13: How would you go about describing the open source projects like JUnit (unit testing), Ant (build tool), CVS (version control system) and log4J (logging tool) which are integral part of most Java/J2EE projects? _____	199
Q 14: How would you go about describing Web services? _____	206
<b>Emerging Technologies/Frameworks... _____</b>	<b>210</b>
Q 01: What is Test Driven Development (TDD)? _____	211
Q 02: What is the point of Test Driven Development (TDD)? _____	211
Q 03: What is aspect oriented programming? Explain AOP? _____	212
Q 04: What are the differences between OOP and AOP? _____	214
Q 05: What are the benefits of AOP? _____	214
Q 06: What is attribute or annotation oriented programming? _____	215
Q 07: What are the pros and cons of annotations over XML based deployment descriptors? _____	215
Q 08: What is XDoclet? _____	216
Q 09: What is inversion of control (IOC) (also known as dependency injection)? _____	216
Q 10: What are the different types of dependency injections? _____	217
Q 11: What are the benefits of IOC (aka Dependency Injection)? _____	217
Q 12: What is the difference between a service locator pattern and an inversion of control pattern? _____	217
Q 13: Why dependency injection is more elegant than a JNDI lookup to decouple client and the service? _____	218
Q 14: Explain Object-to-Relational (O/R) mapping? _____	218
Q 15: Give an overview of hibernate framework? _____	218
Q 16: Explain some of the pitfalls of Hibernate and explain how to avoid them? _____	220
Q 17: Give an overview of the Spring framework? _____	221
Q 18: How would EJB 3.0 simplify your Java development compared to EJB 1.x, 2.x? _____	222
Q 19: Briefly explain key features of the JavaServer Faces (JSF) framework? _____	223
Q 20: How would the JSF framework compare with the Struts framework? _____	225
<b>Sample interview questions... _____</b>	<b>226</b>
Java _____	227
Web components _____	227
Enterprise _____	227
Design _____	229
General _____	229
<b>GLOSSARY OF TERMS _____</b>	<b>230</b>
<b>RESOURCES _____</b>	<b>232</b>
<b>INDEX _____</b>	<b>234</b>

### What this book will do for you?

Have you got the time to read 10 or more books and articles to add value prior to the interview? This book has been written mainly from the perspective of **Java/J2EE job seekers** and **interviewers**. There are numerous books and articles on the market covering specific topics like Java, J2EE, EJB, Design Patterns, ANT, CVS, Multi-Threading, Servlets, JSP, emerging technologies like AOP (Aspect Oriented Programming), Test Driven Development (TDD), Inversion of Control (IoC) etc. But from an interview perspective it is not possible to brush up on all these books where each book usually has from 300 pages to 600 pages. The basic purpose of this book is to cover all the core concepts and design/coding issues which, all Java/J2EE developers, designers and architects should be conversant with to perform well in their current jobs and to launch a successful career by doing well at interviews. The interviewer can also use this book to make sure that they hire the right candidate depending on their requirements. This book contains a wide range of topics relating to Java/J2EE development in a concise manner supplemented with diagrams, tables, sample codes and examples. This book is also appropriately categorised to enable you to choose the area of interest to you.

This book will assist all Java/J2EE practitioners to become better at what they do. Usually it takes years to understand all the core concepts and design/coding issues when you rely only on your work experience. The best way to fast track this is to read appropriate technical information and proactively apply these in your work environment. It worked for me and hopefully it will work for you as well. I was also at one stage undecided whether to name this book "**Java/J2EE core concepts and solving design/coding issues**" or "**Java/J2EE Job Interview Companion**". The reason I chose "**Java/J2EE Job Interview Companion**" is because these core concepts and design/coding issues helped me to be successful in my interviews and also gave me thumbs up in code reviews.

## Motivation for this book

I started using Java in 1999 when I was working as a junior developer. During those two years as a permanent employee, I pro-actively spent many hours studying the core concepts behind Java/J2EE in addition to my hands on practical experience. Two years later I decided to start contracting. Since I started contracting in 2001, my career had a much-needed boost in terms of contract rates, job satisfaction, responsibility etc. I moved from one contract to another with a view of expanding my skills and increasing my contract rates.

In the last 5 years of contracting, I have worked for 5 different organisations both medium and large on 8 different projects. For each contract I held, on average I attended 6-8 interviews with different companies. In most cases multiple job offers were made and consequently I was in a position to negotiate my contract rates and also to choose the job I liked based on the type of project, type of organisation, technology used, etc. I have also sat for around 10 technical tests and a few preliminary phone interviews.

The success in the interviews did not come easily. I spent hours prior to each set of interviews wading through various books and articles as a preparation. The motivation for this book was to collate all this information into a single book, which will save me time prior to my interviews but also can benefit others in their interviews. What is in this book has helped me to go from **just a Java/J2EE job to a career in Java/J2EE** in a short time. It has also given me the job security that 'I can find a contract/permanent job opportunity even in the difficult job market'.

I am not suggesting that every one should go contracting but by performing well at the interviews you can be in a position to pick the permanent role you like and also be able to negotiate your salary package. Those of you who are already in good jobs can impress your team leaders, solution designers and/or architects for a possible promotion by demonstrating your understanding of the key areas discussed in this book. You can discuss with your senior team members about **performance issues, transactional issues, threading issues (concurrency issues) and memory issues**. In most of my previous contracts I was in a position to impress my team leads and architects by pinpointing some of the critical performance, memory, transactional and threading issues with the code and subsequently fixing them. Trust me it is not hard to impress someone if you understand the key areas.

### For example:

- Struts action classes are not thread-safe (Refer **Q113** in Enterprise section).
- JSP variable declaration is not thread-safe (Refer **Q34** in Enterprise section).
- Valuable resources like database connections should be closed properly to avoid any memory and performance issues (Refer **Q45** in Enterprise section).
- Throwing an application exception will not rollback the transaction in EJB. (Refer **Q77** in Enterprise section).

The other key areas, which are vital to any software development, are a good understanding of some of **key design concepts, design patterns**, and a **modelling language** like **UML**. These key areas are really worthy of a mention in your resume and interviews.

### For example:

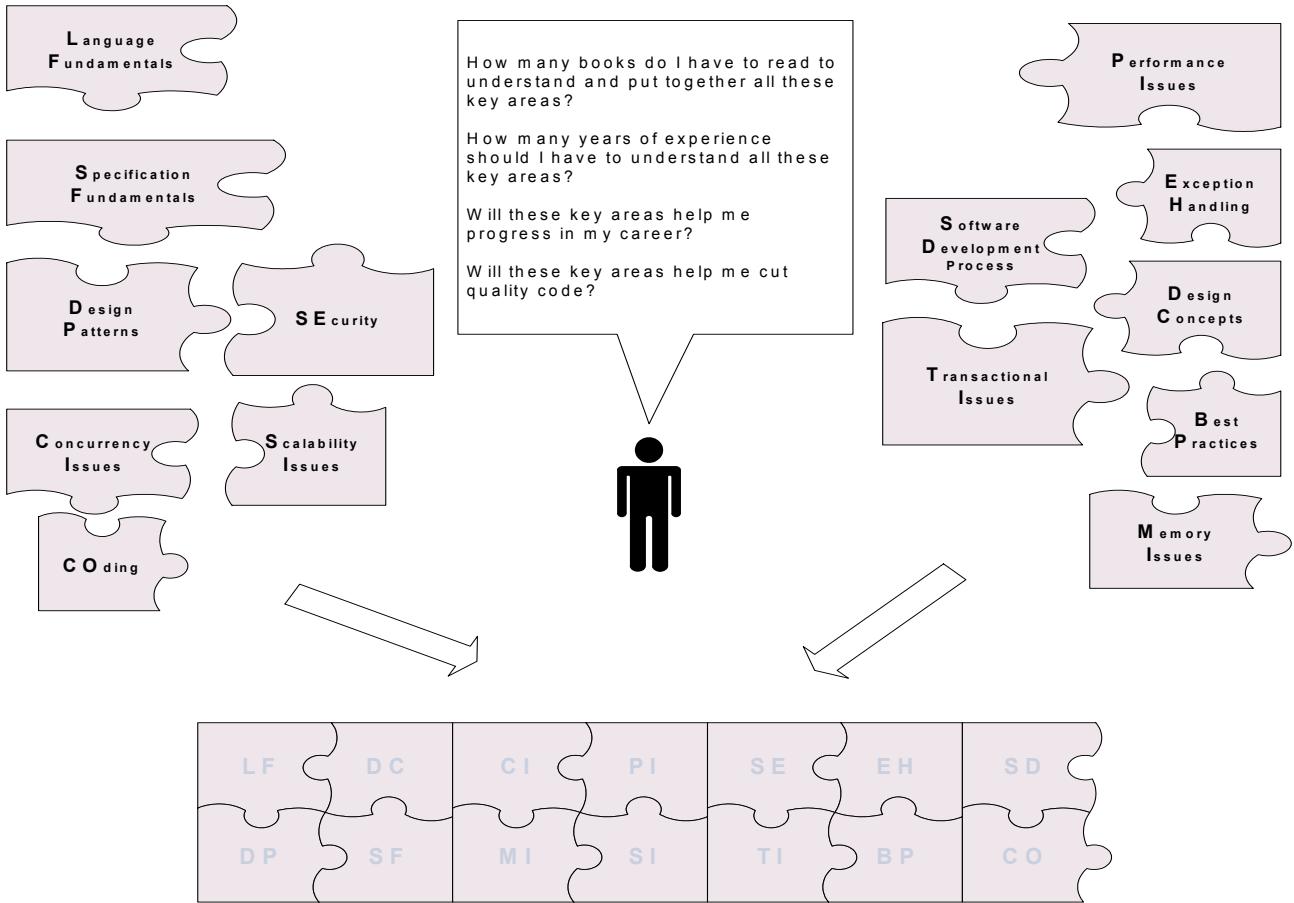
- Know how to use inheritance, polymorphism and encapsulation (Refer **Q5, Q6, Q7, and Q8** in Java section.).
- Why use design patterns? (Refer **Q5** in Enterprise section).
- Why is UML important? (Refer **Q106** in Enterprise section).

If you happen to be in an interview with an organization facing serious issues with regards to their Java application relating to memory leaks, performance problems or a crashing JVM etc then you are likely to be asked questions on these topics. Refer **Q 63 – Q 65** in Java section and **Q123, Q125** in Enterprise section.

Another good reason why these key areas like transactional issues, design concepts, design patterns etc are vital are because solution designers, architects, team leads, and/or senior developers are usually responsible for conducting the technical interviews. These areas are their favourite topics because these are essential to any software development.

Some interviewers request you to write a small program during interview or prior to getting to the interview stage. This is to ascertain that you can code using object oriented concepts and design patterns. So I have included a **coding key area** to illustrate what you need to look for while coding.

- Apply OO concepts like inheritance, polymorphism and encapsulation: Refer **Q08** in Java section.
- Program to interfaces not to implementations: Refer **Q08, Q15** in Java section.
- Use of relevant design patterns: Refer **Q11** in How would you go about... section.
- Use of Java collection API and exceptions correctly: Refer **Q15, Q34**, and **Q35** in Java section.
- Stay away from hard coding values: Refer **Q04** in Java section.



**This book aims to solve the above dilemma.**

My dad keeps telling me to find a permanent job (instead of contracting), which in his view provides better job security but I keep telling him that in my view in Information Technology the job security is achieved only by keeping your knowledge and skills sharp and up to date. The 8 contract positions I held over the last 5.5 years have given me broader experience in Java/J2EE and related technologies. It also kept me motivated since there was always something new to learn in each assignment, and not all companies will appreciate your skills and expertise until you decide to leave. Do the following statements sound familiar to you when you hand in your resignation or decide not to extend your contract after getting another job offer? "Can I tempt you to come back? What can I do to keep you here?" etc. You might even think why you waited so long. The best way to make an impression in any organisations is to understand and proactively apply and resolve the issues relating to the **Key Areas** discussed in the next section. But **be a team player, be tactful and don't be critical of everything, do not act in a superior way and have a sense of humour**.

**"Technical skills must be complemented with interpersonal skills."**

**Quick Read guide:** It is recommended that you go through all the questions in all the sections but if you are pressed for time or would like to read it just before an interview then follow the steps shown below:

1. Read/Browse **Popular Questions** in Java and Enterprise Java sections.
2. Read/Browse **Key Points** in Java and Enterprise Java sections.
3. Read/Browse through "**Emerging Technologies/Frameworks**" section.
4. Read/Browse "**How would you go about...**" section excluding **Q11 & Q13**, which are discussed in detail.

## Key Areas Index

I have categorised the core concepts and issues into **14 key areas** as listed below. These key areas are vital for any good software development. This index will enable you to refer to the questions based on **key areas**. Also note that each question has an icon next to it to indicate which key area or areas it belongs to. Additional reading is recommended for beginners in each of the key areas.

Key Areas	icon	Question Numbers			
		Java section	Enterprise section	How would you go about...?	Emerging Technologies /Frameworks
Language Fundamentals	LF	Q1-Q4, Q10-Q14, Q16-Q20, Q22-Q27, Q30-Q33, Q36-Q43, Q47-Q62	-		Q10, Q15, Q17, Q19
Specification Fundamentals	SF	-	Q1-Q19, Q26-Q33, Q35-Q38, Q41, Q42, Q44, Q46-Q81, Q89-Q97, Q99, 102, Q110, Q112-Q115, Q118-Q119, Q121, Q126, Q127, Q128	Q14	
Design Concepts	DC	Q5-Q9, Q10, Q13, Q22, Q49	Q2, Q3, Q19, Q20, Q21, Q31, Q45, Q98, Q106, Q107, Q108, Q109, 101, Q111	Q02, Q08, Q09	Q3-Q9, Q11, Q13, Q14, Q16, Q18, Q20
Design Patterns	DP	Q10, Q14, Q20, Q31, Q45, Q46, Q50, Q54, Q66	Q5, Q5, Q22, Q24, Q25, Q83, Q84, Q85, Q86, Q87, Q88, Q110, Q111, Q116	Q11	Q12
Transactional Issues	TI	-	Q43, Q71, Q72, Q73, Q74, Q75, Q77, Q78, Q79	Q7	
Concurrency Issues	CI	Q13, Q15, Q29, Q36, Q40, Q53	Q16, Q34, Q113	Q6	
Performance Issues	PI	Q13, Q15 -Q22, Q40, Q53, Q63.	Q10, Q16, Q43, Q45, Q46, Q72, Q83-Q88, Q97, Q98, Q100, Q102, Q123, Q125, Q128	Q3, Q5	
Memory Issues	MI	Q22, Q29, Q32, Q33, Q36, Q45, Q64, Q65.	Q45, Q93	Q3, Q4	
Scalability Issues	SI	Q19, Q20	Q20, Q21, Q120, Q122		
Exception Handling	EH	Q34,Q35	Q76, Q77		
Security	SE	Q61	Q12, Q13, Q23, Q35, Q46, Q51, Q58, Q81	Q12	
Best Practices	BP	Q15, Q21, Q34, Q63, Q64	Q10, Q16, Q39, Q40, Q46, Q82, Q124, Q125		
Software Development Process	SD	-	Q103-Q109, Q129, Q133, Q134, Q136	Q1, Q10, Q13	Q1, Q2
Coding <sup>1</sup>	CO	Q04, Q08, Q10, Q12, Q13, Q15, Q16, Q17, Q21, Q34, Q45, Q46	Q10, Q18, Q21, Q23, Q36, Q38, Q42, Q43, Q45, Q74, Q75, Q76, Q77, Q112, Q114, Q127, Q128	Q11	

<sup>1</sup> Some interviewers request you to write a small program during interview or prior to getting to the interview stage. This is to ascertain that you can code using object oriented concepts and design patterns. I have included a coding key area to illustrate what you need to look for while coding. Unlike other key areas, the CO is not always shown against the question but shown above the actual section of relevance within a question.

## SECTION ONE

### Java – Interview questions & answers

K  
E  
Y  
  
A  
R  
E  
A  
S

- Language Fundamentals **LF**
- Design Concepts **DC**
- Design Patterns **DP**
- Concurrency Issues **CI**
- Performance Issues **PI**
- Memory Issues **MI**
- Exception Handling **EH**
- Security **SE**
- Scalability Issues **SI**
- Coding<sup>1</sup> **CO**

**Popular Questions:** Q01, Q04, Q07, Q08, Q13, Q16, Q17, Q18, Q19, Q25, Q27, Q29, Q32, Q34, Q40, Q45, Q46, Q50, Q51, Q53, Q54, Q55, Q63, Q64, Q66, **Q67**

<sup>1</sup> Unlike other key areas, the **CO** is not always shown against the question but shown above the actual subsection of relevance within a question.

## Java – Language Fundamentals

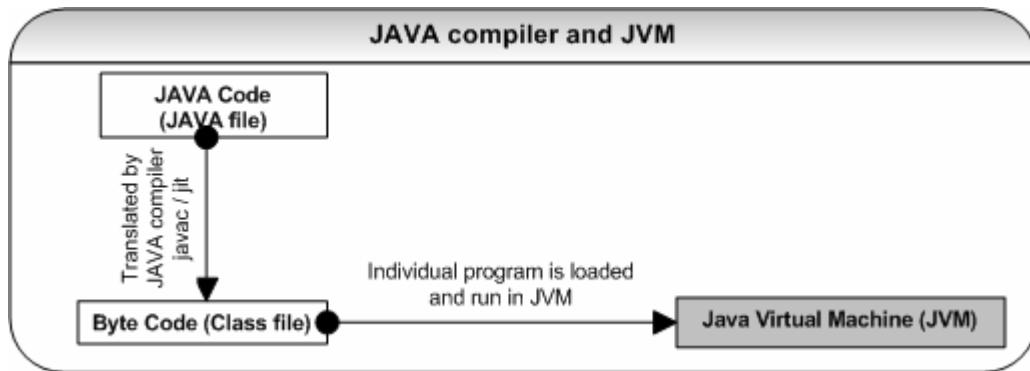
**Q 01:** Give a few reasons for using Java? **LF DC**

**A 01:** Java is a fun language. Let's look at some of the reasons:

- Built-in support for multi-threading, socket communication, and memory management (automatic garbage collection).
- Object Oriented (OO).
- Better portability than other languages across operating systems.
- Supports Web based applications (Applet, Servlet, and JSP), distributed applications (sockets, RMI, EJB etc) and network protocols (HTTP, JRMP etc) with the help of extensive standardised APIs (Application Program Interfaces).

**Q 02:** What is the main difference between the Java platform and the other software platforms? **LF**

**A 02:** Java platform is a software-only platform, which runs on top of other hardware-based platforms like UNIX, NT etc.



The Java platform has 2 components:

- Java Virtual Machine (**JVM**) – 'JVM' is a software that can be ported onto various hardware platforms. Byte codes are the machine language of the JVM.
- Java Application Programming Interface (**Java API**) -

**Q 03:** What is the difference between C++ and Java? **LF**

**A 03:** Both C++ and Java use similar syntax and are Object Oriented, but:

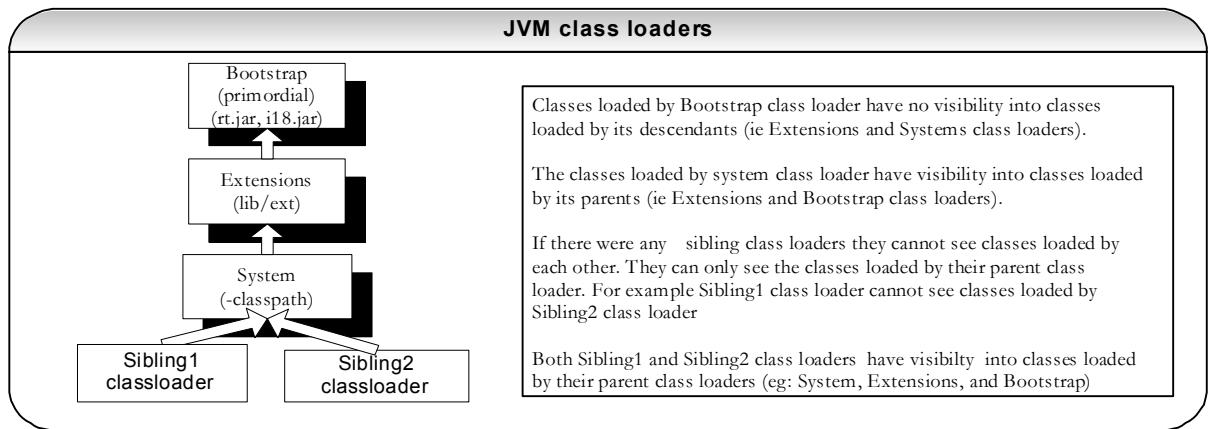
- Java does not support pointers. Pointers are inherently tricky to use and troublesome.
- Java does not support multiple inheritances because it causes more problems than it solves. Instead Java supports **multiple interface inheritance**, which allows an object to inherit many method signatures from different interfaces with the condition that the inheriting object must implement those inherited methods. The multiple interface inheritance also allows an object to behave **polymorphically** on those methods. [Refer **Q 8** and **Q 10** in Java section.]
- Java does not support destructors but rather adds a `finalize()` method. Finalize methods are invoked by the garbage collector prior to reclaiming the memory occupied by the object, which has the `finalize()` method. This means you do not know when the objects are going to be finalized. **Avoid using finalize() method to release non-memory resources** like file handles, sockets, database connections etc because Java has only a finite number of these resources and you do not know when the garbage collection is going to kick in to release these resources through the `finalize()` method.
- Java does not include structures or unions because the traditional data structures are implemented as an object oriented framework (Java collection framework – Refer **Q14, Q15** in Java section).

- All the code in Java program is encapsulated within classes therefore Java does not have global variables or functions.
- C++ requires explicit memory management, while Java includes automatic garbage collection. [Refer **Q32** in Java section].

**Q 04:** Explain Java class loaders? Explain dynamic class loading? **[LF]**

**A 04:** Class loaders are hierarchical. Classes are introduced into the JVM as they are referenced by name in a class that is **already running** in the JVM. So how is the very first class loaded? The very first class is specially loaded with the help of *static main()* method declared in your class. All the subsequently loaded classes are loaded by the classes, which are already loaded and running. A class loader creates a namespace. All JVMs include at least one class loader that is embedded within the JVM called the primordial (or bootstrap) class loader. Now let's look at non-primordial class loaders. The JVM has hooks in it to allow user defined class loaders to be used in place of primordial class loader. Let us look at the class loaders created by the JVM.

CLASS LOADER	reloadable?	Explanation
Bootstrap (primordial)	No	Loads JDK internal classes, <i>java.*</i> packages. (as defined in the sun.boot.class.path system property, typically loads rt.jar and i18n.jar)
Extensions	No	Loads jar files from JDK extensions directory (as defined in the java.ext.dirs system property – usually lib/ext directory of the JRE)
System	No	Loads classes from system classpath (as defined by the java.class.path property, which is set by the <b>CLASSPATH</b> environment variable or –classpath or –cp command line options)



Class loaders are hierarchical and use a **delegation model** when loading a class. Class loaders request their parent to load the class first before attempting to load it themselves. When a class loader loads a class, the child class loaders in the hierarchy will never reload the class again. Hence **uniqueness** is maintained. Classes loaded by a child class loader have **visibility** into classes loaded by its parents up the hierarchy but the reverse is not true as explained in the above diagram.

**Important:** Two objects loaded by different class loaders are never equal even if they carry the same values, which mean a class is uniquely identified in the context of the associated class loader. This applies to **singletons** too, where **each class loader will have its own singleton**. [Refer **Q45** in Java section for singleton design pattern]

**Explain static vs. dynamic class loading?**

Static class loading	Dynamic class loading
Classes are statically loaded with Java's "new" operator.	Dynamic loading is a technique for programmatically invoking the functions of a class loader at run time. Let us look at how to load classes dynamically.
<pre>class MyClass {     public static void main(String args[]) {         Car c = new Car();     } }</pre>	<b>Class.forName (String className);</b> //static method which returns a Class  The above static method returns the class object associated with the class name. The string <i>className</i> can be supplied dynamically at run time. Unlike the static loading, the dynamic loading will decide whether to load the class <i>Car</i> or the class <i>Jeep</i> at runtime based on a properties file and/or other runtime

	<p>conditions. Once the class is dynamically loaded the following method returns an instance of the loaded class. It's just like creating a class object with no arguments.</p> <pre><b>class.newInstance ()</b>; //A non-static method, which creates an instance of a class (i.e. creates an object).</pre> <pre>Jeep myJeep = null; //myClassName should be read from a properties file or Constants interface. //stay away from hard coding values in your program. <b>CO</b> String myClassName = "au.com.Jeep"; Class vehicleClass = <b>Class.forName</b>(myClassName); myJeep = (Jeep) vehicleClass.<b>newInstance</b>(); myJeep.setFuelCapacity(50);</pre>
A <b>NoClassDefFoundException</b> is thrown if a class is referenced with Java's "new" operator (i.e. static loading) but the runtime system cannot find the referenced class.	<p>A <b>ClassNotFoundException</b> is thrown when an application tries to load in a class through its string name using the following methods but no definition for the class with the specified name could be found:</p> <ul style="list-style-type: none"> <li>▪ The <b>forName(..)</b> method in class - <b>Class</b>.</li> <li>▪ The <b>findSystemClass(..)</b> method in class - <b>ClassLoader</b>.</li> <li>▪ The <b>loadClass(..)</b> method in class - <b>ClassLoader</b>.</li> </ul>

**What are “static initializers” or “static blocks with no function names”?** When a class is loaded, all blocks that are declared static and don't have function name (i.e. static initializers) are executed even before the constructors are executed. As the name suggests they are typically used to initialize static fields. **CO**

```
public class StaticInitilaizer {
 public static final int A = 5;
 public static final int B;

 //Static initializer block, which is executed only once when the class is loaded.

 static {
 if(A == 5)
 B = 10;
 else
 B = 5;
 }

 public StaticInitilaizer(){} // constructor is called only after static initializer block
}
```

The following code gives an **Output of A=5, B=10**.

```
public class Test {
 System.out.println("A =" + StaticInitilaizer.A + ", B =" + StaticInitilaizer.B);
}
```

**Q 05:** What are the advantages of Object Oriented Programming Languages (OOPL)? **DC**

**A 05:** The Object Oriented Programming Languages directly represent the real life objects like *Car, Jeep, Account, Customer* etc. The features of the OO programming languages like **polymorphism, inheritance and encapsulation** make it powerful. [Tip: remember **pie** which, stands for Polymorphism, Inheritance and Encapsulation are the **3 pillars** of OOPL]

**Q 06:** How does the Object Oriented approach improve software development? **DC**

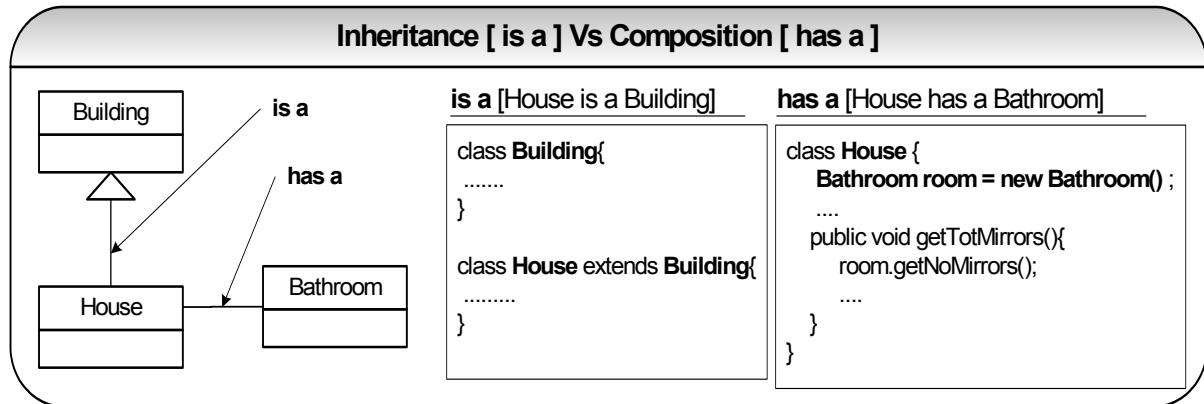
**A 06:** The key benefits are:

- **Re-use** of previous work: using **implementation inheritance** and **object composition**.
- **Real mapping to the problem domain:** Objects map to real world and represent vehicles, customers, products etc: with **encapsulation**.
- **Modular Architecture:** Objects, systems, frameworks etc are the building blocks of larger systems.

The **increased quality** and **reduced development time** are the by-products of the key benefits discussed above. If 90% of the new application consists of proven existing components then only the remaining 10% of the code have to be tested from scratch.

**Q 07:** How do you express an '**is a**' relationship and a '**has a**' relationship or explain inheritance and composition? What is the difference between composition and aggregation? **[DC]**

**A 07:** The '**is a**' relationship is expressed with **inheritance** and '**has a**' relationship is expressed with **composition**. Both inheritance and composition allow you to place sub-objects inside your new class. Two of the main techniques for **code reuse** are **class inheritance** and **object composition**.



**Inheritance** is uni-directional. For example *House is a Building*. But *Building* is not a *House*. Inheritance uses **extends** key word. **Composition:** is used when *House has a Bathroom*. It is incorrect to say *House is a Bathroom*. Composition simply means using instance variables that refer to other objects. The class *House* will have an instance variable, which refers to a *Bathroom* object.

**Which one to use?** The guide is that inheritance should be only used when *subclass 'is a' superclass*.

- Don't use inheritance just to get code reuse. If there is no '**is a**' relationship then use composition for code reuse. Overuse of **implementation inheritance** (uses the "extends" key word) can break all the subclasses, if the superclass is modified.
- Do not use inheritance just to get polymorphism. If there is no '**is a**' relationship and all you want is **polymorphism** then use **interface inheritance** with **composition**, which gives you **code reuse** (Refer **Q8** in Java section for interface inheritance).

**What is the difference between aggregation and composition?**

Aggregation	Composition
Aggregation is an association in which one class belongs to a collection. This is a part of a whole relationship where a part can exist without a whole. <b>For example</b> a line item is a whole and product is a part. If a line item is deleted then corresponding product need not be deleted. So <b>aggregation has a weaker relationship</b> .	Composition is an association in which one class belongs to a collection. This is a part of a whole relationship where a part cannot exist without a whole. If a whole is deleted then all parts are deleted. <b>For example</b> An order is a whole and line items are parts. If an order deleted then all corresponding line items for that order should be deleted. So <b>composition has a stronger relationship</b> .

**Q 08:** What do you mean by polymorphism, inheritance, encapsulation, and dynamic binding? **[DC]**

**A 08:** **Polymorphism** – means the ability of a single variable of a given type to be used to reference objects of different types, and automatically call the method that is specific to the type of object the variable references. In a nutshell, polymorphism is a bottom-up method call. The benefit of polymorphism is that it is **very easy to add new classes of derived objects without breaking the calling code** (i.e. `getTotArea()` in the sample code shown below) that uses the polymorphic classes or interfaces. When you send a message to an object even though you don't know what specific type it is, and the right thing happens, that's called **polymorphism**. The process used by object-oriented programming languages to implement polymorphism is called **dynamic binding**. Let us look at some sample code to demonstrate polymorphism: **[CO]**

**Sample code:**

```

//client or calling code
double dim = 5.0; //ie 5 meters radius or width
List listShapes = new ArrayList(20);

Shape s = new Circle();
listShapes.add(s); //add circle

s = new Square();
listShapes.add(s); //add square

getTotArea (listShapes,dim); //returns 78.5+25.0=103.5

//Later on, if you decide to add a half circle then define
//a HalfCircle class, which extends Circle and then provide an
//area(). method but your called method getTotArea(...) remains
//same.

s = new HalfCircle();
listShapes.add(s); //add HalfCircle

getTotArea (listShapes,dim); //returns 78.5+25.0+39.25=142.75

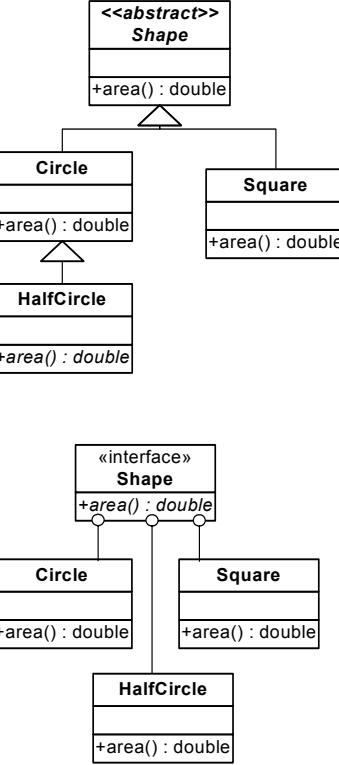
/** called method: method which adds up areas of various
** shapes supplied to it.
*/
public double getTotArea(List listShapes, double dim){
 Iterator it = listShapes.iterator();
 double totalArea = 0.0;
 //loop through different shapes
 while(it.hasNext()){
 Shape s = (Shape) it.next();
 totalArea += s.area(dim);
 }
 return totalArea ;
}

```

**For example:** given a base class/interface *Shape*, polymorphism allows the programmer to define different area(double dim) methods for any number of derived classes such as *Circle*, *Square* etc. No matter what shape an object is, applying the area method to it will return the right results.

Later on *HalfCircle* can be added without breaking your called code i.e. method getTotArea(...)

Depending on what the shape is, appropriate area(double dim) method gets called and calculated.  
*Circle* → area is 78.5sqm  
*Square* → area is 25sqm  
*HalfCircle* → area is 39.25 sqm



**Inheritance** – is the inclusion of behaviour (i.e. methods) and state (i.e. variables) of a base class in a derived class so that they are accessible in that derived class. The key benefit of Inheritance is that it provides the formal mechanism for **code reuse**. Any shared piece of business logic can be moved from the derived class into the base class as part of refactoring process to improve maintainability of your code by avoiding code duplication. The existing class is called the *superclass* and the derived class is called the *subclass*. **Inheritance** can also be defined as the process whereby one object acquires characteristics from one or more other objects the same way children acquire characteristics from their parents.

There are two types of inheritances:

**1. Implementation inheritance** (aka class inheritance): You can extend an applications' functionality by reusing functionality in the parent class by inheriting all or some of the operations already implemented. In Java, you can only inherit from one superclass. Implementation inheritance promotes reusability but improper use of class inheritance can cause programming nightmares by breaking encapsulation and making future changes a problem. With implementation inheritance, the subclass becomes tightly coupled with the superclass. This will make the design fragile because if you want to change the superclass, you must know all the details of the subclasses to avoid breaking them. So when using implementation inheritance, **make sure that the subclasses depend only on the behaviour of the superclass, not on the actual implementation**. For example in the above diagram the subclasses should only be concerned about the behaviour known as `area()` but not how it is implemented.

**2. Interface inheritance** (aka type inheritance): This is also known as subtyping. Interfaces provide a mechanism for specifying a relationship between otherwise unrelated classes, typically by specifying a set of common methods each implementing class must contain. Interface inheritance promotes the design concept of **program to interfaces not to implementations**. This also reduces the coupling or implementation dependencies between systems. In Java, you can implement any number of interfaces. This is more flexible than implementation inheritance because it won't lock you into specific implementations which make subclasses difficult to maintain. So care should be taken not to break the implementing classes by modifying the interfaces.

**Which one to use?** Prefer interface inheritance to implementation inheritance because it promotes the design concept of **coding to an interface** and **reduces coupling**. Interface inheritance can achieve **code reuse** with the help of **object composition**. If you look at Gang of Four (GoF) design patterns, you can see that it favours interface inheritance to implementation inheritance. **CO**

Implementation inheritance	Interface inheritance
<p>Let's assume that savings account and term deposit account have a similar behaviour in terms of depositing and withdrawing money, so we will get the super class to implement this behaviour and get the subclasses to reuse this behaviour. But saving account and term deposit account have specific behaviour in calculating the interest.</p> <pre data-bbox="238 375 801 1051"> public abstract class Account {     public void deposit(double amount) {         //deposit logic     }      public void withdraw(double amount) {         //withdraw logic     }      public abstract double calculateInterest(double amount); }  public class SavingsAccount extends Account {     public double calculateInterest(double amount) {         //calculate interest for SavingsAccount     } }  public class TermDepositAccount extends Account {     public double calculateInterest(double amount) {         //calculate interest for TermDeposit     } }</pre> <p>The calling code can be defined as follows for illustration purpose only:</p> <pre data-bbox="238 1157 801 1495"> public class Test {     public static void main(String[] args) {         Account acc1 = new SavingsAccount();         acc1.deposit(5.0);         acc1.withdraw(2.0);          Account acc2 = new TermDepositAccount();         acc2.deposit(10.0);         acc2.withdraw(3.0);          acc1.calculateInterest(500.00);         acc2.calculateInterest(500.00);     } }</pre>	<p>Let's look at an <b>interface inheritance</b> code sample, which makes use of <b>composition</b> for reusability. In the following example the methods <code>deposit(...)</code> and <code>withdraw(...)</code> share the same piece of code in <code>AccountHelper</code> class. The method <code>calculateInterest(...)</code> has its specific implementation in its own class.</p> <pre data-bbox="817 354 1396 671"> public interface Account {     public abstract void deposit(double amount);     public abstract void withdraw(double amount);     public abstract int getAccountType(); }  public interface SavingsAccount extends Account{     public abstract double calculateInterest(double amount); }  public interface TermDepositAccount extends Account{     public abstract double calculateInterest(double amount); }</pre> <p>The classes <code>SavingsAccountImpl</code>, <code>TermDepositAccountImpl</code> should implement the methods declared in its interfaces. The class <code>AccountHelper</code> implements the methods <code>deposit(...)</code> and <code>withdraw(...)</code></p> <pre data-bbox="817 819 1462 1305"> public class SavingsAccountImpl implements SavingsAccount{     private int accountType = 1;      //helper class which promotes code reuse through composition     AccountHelper helper = new AccountHelper();      public void deposit(double amount) {         helper.deposit(amount, getAccountType());     }      public void withdraw(double amount) {         helper.withdraw(amount, getAccountType());     }      public double calculateInterest(double amount) {         //calculate interest for SavingsAccount     }      public int getAccountType(){         return accountType;     } }</pre> <pre data-bbox="817 1315 1429 1801"> public class TermDepositAccountImpl implements         TermDepositAccount {     private int accountType = 2;      //helper class which promotes code reuse through composition     AccountHelper helper = new AccountHelper();      public void deposit(double amount) {         helper.deposit(amount, getAccountType());     }      public void withdraw(double amount) {         helper.withdraw(amount, getAccountType());     }      public double calculateInterest(double amount) {         //calculate interest for TermDeposit     }      public int getAccountType() {         return accountType;     } }</pre> <p>The calling code can be defined as follows for illustration purpose only:</p> <pre data-bbox="817 1896 1201 1953"> public class Test {     public static void main(String[] args) {</pre>

```

Account acc1 = new SavingsAccountImpl();
acc1.deposit(5.0);

Account acc2 = new TermDepositAccountImpl();
acc2.deposit(10.0);

if (acc1.getAccountType() == 1) {
 ((SavingsAccount) acc1).calculateInterest(500.00);
}

if (acc2.getAccountType() == 2) {
 ((TermDepositAccount) acc2).calculateInterest(500.00);
}
}

```

**Encapsulation** – refers to keeping all the related members (variables and methods) together in an object. Specifying members as private can hide the variables and methods. Objects should hide their inner workings from the outside view. Good **encapsulation improves code modularity by preventing objects interacting with each other in an unexpected way**, which in turn makes future development and refactoring efforts easy.

Being able to encapsulate members of a class is important for **security** and **integrity**. We can protect variables from unacceptable values. The sample code below describes how encapsulation can be used to protect the **MyMarks** object from having negative values. Any modification to member variable “**vmarks**” can only be carried out through the setter method **setMarks(int mark)**. This prevents the object “**MyMarks**” from having any negative values by throwing an exception. **CO**

#### Sample code

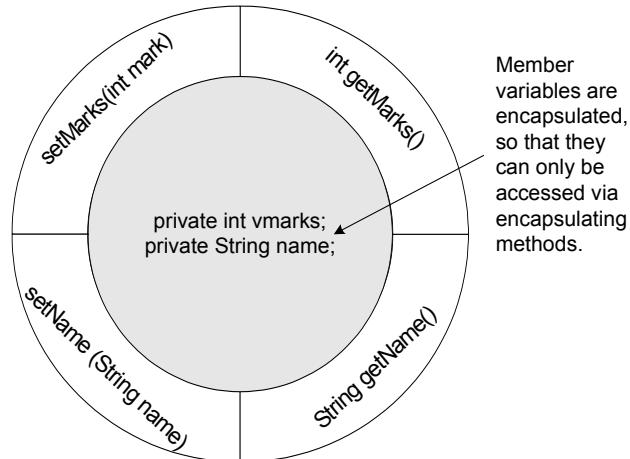
```

Class MyMarks {
 private int vmarks = 0;
 private String name;

 public void setMarks(int mark)
 throws MarkException {
 if(mark > 0)
 this.vmarks = mark;
 else {
 throw new MarkException("No negative
 Values");
 }
 }

 public int getMarks(){
 return vmarks;
 }
 //getters and setters for attribute name goes here.
}

```



**Q 09:** What is design by contract? Explain the **assertion** construct? **DC**

**A 09:** Design by contract specifies the obligations of a calling-method and called-method to each other. Design by contract is a valuable technique, which should be used to build well-defined interfaces. The strength of this programming methodology is that it gets the programmer to **think clearly about what a function does**, what pre and post conditions it must adhere to and also it **provides documentation for the caller**. Java uses the **assert** statement to implement pre- and post-conditions. Java's exceptions handling also support design by contract especially **checked exceptions** (Refer **Q34** in Java section for checked exceptions). In design by contract in addition to specifying programming code to carrying out intended operations of a method the programmer also specifies:

**1] Preconditions** – This is the part of the contract the **calling-method must agree to**. Preconditions specify the conditions that must be true before a called method can execute. Preconditions involve the system state and the arguments passed into the method at the time of its invocation. **If a precondition fails then there is a bug in the calling-method or calling software component.**

On public methods	On non-public methods
<p><b>Preconditions</b> on <i>public</i> methods are enforced by explicit checks that throw particular, specified exceptions. You <b>should not use assertion to check the parameters of the public methods</b> but can use for the non-public methods. <b>Assert</b> is inappropriate because the method guarantees that it will always enforce the argument checks. It must check its arguments whether or not assertions are enabled. Further, assert construct does not throw an exception of a specified type. It can throw only an <i>AssertionError</i>.</p> <pre data-bbox="277 454 926 601"><code>public void setRate(int rate) {     if(rate &lt;= 0    rate &gt; MAX_RATE){         throw new IllegalArgumentException("Invalid rate → " + rate);     }     setCalculatedRate(rate); }</code></pre>	<p>You can use assertion to check the parameters of the non-public methods.</p> <pre data-bbox="926 285 1488 380"><code>private void setCalculatedRate(int rate) {     assert (rate &gt; 0 &amp;&amp; rate &lt; MAX_RATE) : rate;     //calculate the rate and set it. }</code></pre> <p>Assertions can be disabled, so programs must not assume that assert construct will be always executed:</p> <pre data-bbox="926 475 1488 559"><code>//Wrong: if assertion is disabled, CarpenterJob never //Get removed assert jobsAd.remove(PilotJob);</code></pre> <p><b>Correct:</b></p> <pre data-bbox="926 601 1488 654"><code>boolean pilotJobRemoved = jobsAd.remove(PilotJob); assert pilotJobRemoved;</code></pre>

**2. Postconditions** – This is the part of the contract the **called-method agrees to**. What must be true after a method completes successfully. Postconditions can be used with assertions in both public and non-public methods. The postconditions involve the old system state, the new system state, the method arguments and the method's return value. **If a postcondition fails then there is a bug in the called-method or called software component.**

```
public double calcRate(int rate) {
 if(rate <= 0 || rate > MAX_RATE){
 throw new IllegalArgumentException("Invalid rate !!! ");
 }

 //logic to calculate the rate and set it goes here

 assert this.evaluate(result) < 0 : this; //this → message sent to AssertionError on failure
 return result;
}
```

**3. Class invariants** - what must be true about each instance of a class? A class invariant as an internal invariant that can specify the relationships among multiple attributes, and should be true before and after any method completes. **If an invariant fails then there could be a bug in either calling-method or called-method**. There is no particular mechanism for checking invariants but it is convenient to combine all the expressions required for checking invariants into a single internal method that can be called by assertions. For example if you have a class, which deals with negative integers then you define the **isNegative()** convenient internal method:

```
class NegativeInteger {
 Integer value = new Integer (-1); //invariant

 //constructor
 public NegativeInteger(Integer int) {
 //constructor logic goes here
 assert isNegative();
 }

 //rest of the public and non-public methods goes here. public methods should call assert isNegative(); prior to its return

 //convenient internal method for checking invariants. Returns true if the integer value is negative
 private boolean isNegative(){
 return value.intValue() < 0 ;
 }
}
```

The **isNegative()** method should be true before and after any method completes, each public method and constructor should contain the following assert statement immediately prior to its return.

```
assert isNegative();
```

**Explain the assertion construct?** The assertion statements have two forms as shown below:

```
assert Expression1;
```

```
assert Expression1 : Expression2;
```

Where:

- **Expression1** → is a boolean expression. If the **Expression1** evaluates to false, it throws an *AssertionError* without any detailed message.
- **Expression2** → if the **Expression1** evaluates to false throws an *AssertionError* with using the value of the **Expression2** as the errors' detailed message.

**Note:** If you are using assertions (available from JDK1.4 onwards), you should supply the JVM argument to enable it by package name or class name.

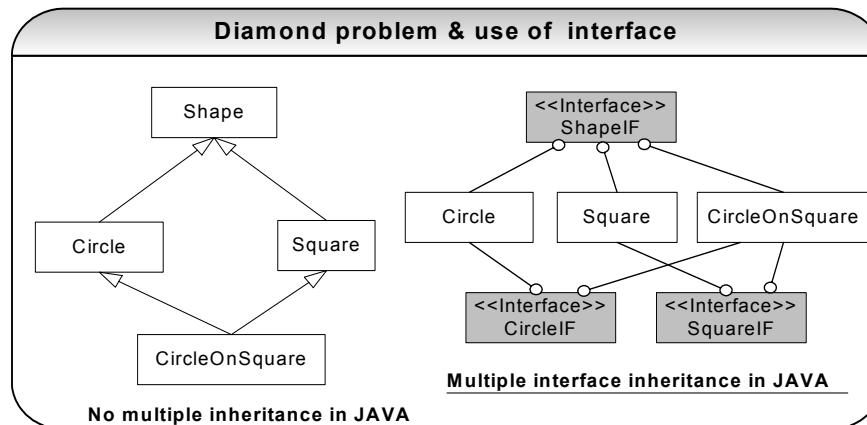
```
Java -ea[:packagename...]:classname] or Java -enableassertions[:packagename...]:classname]
Java -ea:Account
```

**Q 10:** What is the difference between an abstract class and an interface and when should you use them? **LF DP DC**

**A 10:** In design, you want the base class to present *only* an interface for its derived classes. This means, you don't want anyone to actually instantiate an object of the base class. You only **want to upcast to it** (implicit upcasting, which gives you polymorphic behaviour), so that its interface can be used. This is accomplished by making that class **abstract** using the **abstract** keyword. If anyone tries to make an object of an **abstract** class, the compiler prevents it.

The **interface** keyword takes this concept of an **abstract** class a step further by preventing any method or function implementation at all. You can only declare a method or function but not provide the implementation. The class, which is implementing the interface, should provide the actual implementation. The **interface** is a very useful and commonly used aspect in OO design, as it provides the **separation of interface and implementation** and enables you to:

- Capture similarities among unrelated classes without artificially forcing a class relationship.
- Declare methods that one or more classes are expected to implement.
- Reveal an object's programming interface without revealing its actual implementation.
- Model multiple interface inheritance in Java, which provides some of the benefits of full on multiple inheritances, a feature that some object-oriented languages support that allow a class to have more than one superclass.



Abstract class	Interface
Have executable methods and abstract methods.	Have no implementation code. All methods are abstract.
Can only subclass one abstract class.	A class can implement any number of interfaces.
Can have instance variables, constructors and any visibility: public, private, protected, none (aka package).	Cannot have instance variables, constructors and can have only public and none (aka package) visibility.

**When to use an abstract class?**: In case where you want to use **implementation inheritance** then it is usually provided by an abstract base class. Abstract classes are excellent candidates inside of application frameworks. Abstract classes let you define some default behaviour and force subclasses to provide any specific behaviour. Care should be taken not to overuse implementation inheritance as discussed in **Q8** in Java section.

**When to use an interface?**: For polymorphic interface inheritance, where the client wants to only deal with a type and does not care about the actual implementation use interfaces. If you need to change your design frequently, you should prefer using interface to abstract. **CO** Coding to an interface **reduces coupling** and interface inheritance can achieve **code reuse** with the help of **object composition**. Another justification for using interfaces is that they solve the '**diamond problem**' of traditional multiple inheritance as shown in the figure. Java does not support multiple inheritances. Java only supports **multiple interface inheritance**. Interface will solve all the ambiguities caused by this 'diamond problem'.

**Design pattern:** Strategy design pattern lets you swap new algorithms and processes into your program without altering the objects that use them. **Strategy design pattern:** Refer **Q11** in How would you go about... section.

**Q 11:** Why there are some interfaces with no defined methods (i.e. marker interfaces) in Java? **LF** **CO**

**A 11:** The interfaces with no defined methods act like markers. They just tell the compiler that the objects of the classes implementing the interfaces with no defined methods need to be treated differently. **Example** Serializable (Refer **Q19** in Java section), Cloneable etc

**Q 12:** When is a method said to be overloaded and when is a method said to be overridden? **LF** **CO**

**A 12:**

Method Overloading	Method Overriding
<p>Overloading deals with multiple methods in the same class with the same name but different method signatures.</p> <pre>class MyClass {     public void getInvestAmount(int rate) {...}      public void getInvestAmount(int rate, long principal)     { ... } }</pre> <p>Both the above methods have the same method names but different method signatures, which mean the methods are overloaded.</p>	<p>Overriding deals with two methods, one in the parent class and the other one in the child class and has the same name and signatures.</p> <pre>class BaseClass{     public void getInvestAmount(int rate) {...} }  class MyClass extends BaseClass {     public void getInvestAmount(int rate) { ... } }</pre> <p>Both the above methods have the same method names and the signatures but the method in the subclass <i>MyClass</i> overrides the method in the superclass <i>BaseClass</i>.</p>

**Q 13:** What is the main difference between an ArrayList and a Vector? What is the main difference between Hashmap and Hashtable? **LF** **DC** **P1** **C1**

**A 13:**

Vector / Hashtable	ArrayList / Hashmap
Original classes before the introduction of Collections API. <i>Vector</i> & <i>Hashtable</i> are synchronized. Any method that touches their contents is thread-safe.	So if you don't need a thread safe collection, use the <i>ArrayList</i> or <i>Hashmap</i> . Why pay the price of synchronization unnecessarily at the expense of performance degradation.

**So which is better?** As a general rule, prefer *ArrayList/Hashmap* to *Vector/Hashtable*. If your application is a multithreaded application and **at least one of the threads either adds or deletes an entry into the collection** then use new Java *collection* API's external synchronization facility as shown below to **temporarily synchronize** your collections as needed: **CO**

```
Map myMap = Collections.synchronizedMap (myMap);
List myList = Collections.synchronizedList (myList);
```

Java arrays are even faster than using an *ArrayList/Vector* and perhaps therefore may be preferable. *ArrayList/Vector* internally uses an array with some convenient methods like *add(..)*, *remove(..)* etc.

**Q 14:** Explain the Java Collection framework? **LF** **DP**

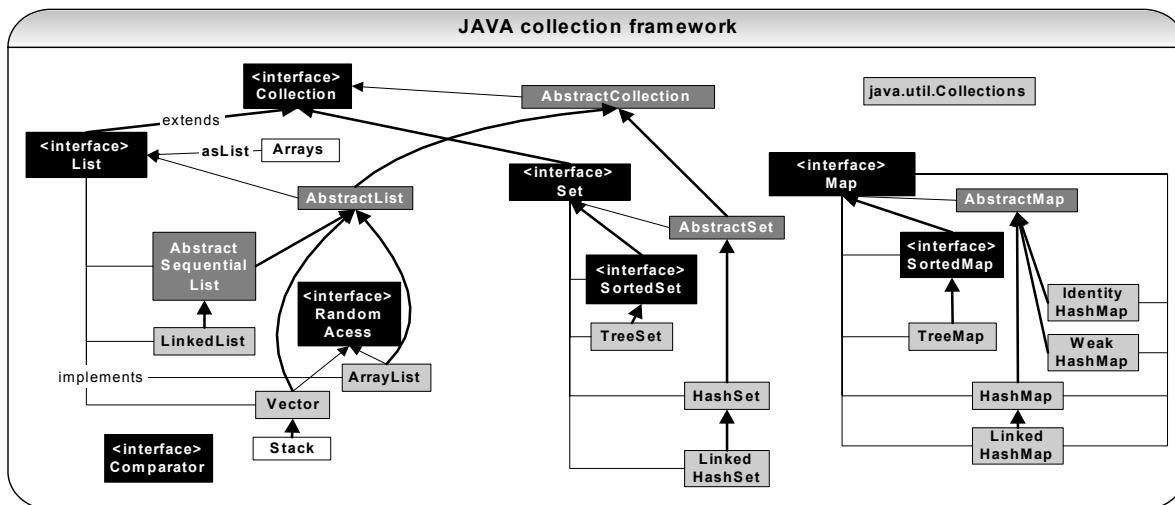
**A 14:** The key interfaces used by the collection framework are **List**, **Set** and **Map**. The **List** and **Set** extends the **Collection** interface. Should not confuse the **Collection** interface with the **Collections** class which is a utility class.

A **Set** is a collection with unique elements and prevents duplication within the collection. **HashSet** and **TreeSet** are implementations of a **Set** interface. A **List** is a collection with an ordered sequence of elements and may contain duplicates. **ArrayList**, **LinkedList** and **Vector** are implementations of a **List** interface.

The Collection API also supports maps, but within a hierarchy distinct from the **Collection** interface. A **Map** is an object that maps keys to values, where the list of keys is itself a collection object. A map can contain duplicate values, but the keys in a map must be distinct. **HashMap**, **TreeMap** and **Hashtable** are implementations of a **Map** interface.

**How to implement collection ordering?** **SortedSet** and **SortedMap** interfaces maintain sorted order. The classes, which implement the **Comparable** interface, impose natural order. For classes that don't implement comparable interface, or when one needs even more control over ordering based on multiple attributes, a **Comparator** interface should be used.

**Design pattern: What is an Iterator?** An Iterator is a use once object to access the objects stored in a collection. **Iterator design pattern** (aka Cursor) is used, which is a behavioural design pattern that provides a way to access elements of a collection sequentially without exposing its internal representation.



(Diagram sourced from: <http://www.wilsonmar.com/1arrays.htm>)

**What are the benefits of the Java collection framework?** Collection framework provides flexibility, performance, and robustness.

- **Polymorphic algorithms** – sorting, shuffling, reversing, binary search etc.
- **Set algebra** - such as finding subsets, intersections, and unions between objects.
- **Performance** - collections have much better performance compared to the older **Vector** and **Hashtable** classes with the elimination of synchronization overheads.
- **Thread-safety** - when synchronization is required, wrapper implementations are provided for temporarily synchronizing existing collection objects.
- **Immutability** - when immutability is required wrapper implementations are provided for making a collection immutable.
- **Extensibility** - interfaces and abstract classes provide an excellent starting point for adding functionality and features to create specialized object collections.

**Q 15:** What are some of the best practices relating to Java collection? **BP** **PI** **CI**

**A 15:**

- Use **ArrayLists**, **HashMap** etc as opposed to **Vector**, **Hashtable** etc, where possible to avoid any synchronization overhead. Even better is to use just arrays where possible. If multiple threads concurrently access a collection and **at least one of the threads either adds or deletes an entry into the collection**, then the collection must be externally synchronized. This is achieved by:

```
Map myMap = Collections.synchronizedMap (myMap);
```

```
List myList = Collections.synchronizedList (myList);
```

- Set the initial capacity of a collection appropriately (e.g. ArrayList, HashMap etc). This is because collection classes like ArrayList, HashMap etc must grow periodically to accommodate new elements. But if you have a very large array, and you know the size in advance then you can speed things up by setting the initial size appropriately.

**For example:** HashMaps/Hashtables need to be created with sufficiently large capacity to minimise **rehashing** (which happens every time the table grows). HashMap has two parameters initial capacity and load factor that affect its performance and space requirements. Higher load factor values (default load factor of 0.75 provides a good trade off between performance and space) will reduce the space cost but will increase the lookup cost of myMap.get(...) and myMap.put(...) methods. When the number of entries in the HashMap exceeds the **current capacity \* loadfactor** then the capacity of the HasMap is roughly doubled by calling the rehash function. It is also very important not to set the initial capacity too high or load factor too low if iteration performance or reduction in space is important.

- Program in terms of interface not implementation:** For example you might decide a LinkedList is the best choice for some application, but then later decide ArrayList might be a better choice for performance reason. **CO**

**Use:**

```
List list = new ArrayList(100); //program in terms of interface & set the initial capacity.
```

**Instead of:**

```
ArrayList list = new ArrayList();
```

- Avoid storing unrelated or different types of objects into same collection:** This is analogous to storing items in pigeonholes without any labelling. To store items use **value objects** or **data objects** (as oppose to storing every attribute in an ArrayList or HashMap). Provide wrapper classes around your collection API classes like ArrayList, Hashmap etc as shown in better approach column. Also where applicable consider using **composite design pattern**, where an object may represent a single object or a collection of objects. Refer **Q52** in Java section for UML diagram of a composite design pattern. **CO**

Avoid where possible	Better approach
<p>The code below is hard to maintain and understand by others. Also gets more complicated as the requirements grow in the future because we are throwing different types of objects like Integer, String etc into a list just based on the indices and it is easy to make mistakes while casting the objects back during retrieval.</p> <pre>List myOrder = new ArrayList() ResultSet rs = ... While (rs.hasNext()) {     List lineItem = new ArrayList();     lineItem.add (new Integer(rs.getInt("itemId")));     lineItem.add (rs.getString("description"));     ...     myOrder.add( lineItem); } return myOrder;</pre> <p><b>Example 2:</b></p> <pre>List myOrder = new ArrayList(10); //create an order OrderVO header = new OrderVO(); header.setOrderId(1001); ... //add all the line items LineItemVO line1 = new LineItemVO(); line1.setLineItemId(1); LineItemVO line2 = new LineItemVO(); Line2.setLineItemId(2);</pre>	<p>When storing items into a collection define value objects as shown below: (<b>VO</b> is an acronym for <b>Value Object</b>).</p> <pre>public class LineItemVO {     private int itemId;     private String productName;      public int getLineItemId(){return accountId ;}     public int getAccountName(){return accountName; }      public void setLineItemId(int accountId ) {         this.accountId = accountId     }     //implement other getter &amp; setter methods }</pre> <p>Now let's define our base wrapper class, which represents an order:</p> <pre>public abstract class Order {     int orderId;     List lineItems = null;      public abstract int countLineItems();     public abstract boolean add(LineItemVO itemToAdd);     public abstract boolean remove(LineItemVO itemToAdd);     public abstract Iterator getIterator();     public int getOrderId(){return this.orderId; } }</pre> <p>Now a specific implementation of our wrapper class:</p> <pre>public class OverseasOrder extends Order {     public OverseasOrder(int inOrderId) {         this.lineItems = new ArrayList(10);         this.orderId = inOrderId;     } }</pre>

<pre>List lineItems = new ArrayList(); lineItems.add(line1); lineItems.add(line2);  //to store objects myOrder.add(order); // index 0 is an OrderVO object myOrder.add(lineItems); //index 1 is a List of line items  //to retrieve objects myOrder.get(0); myOrder.get(1);</pre>	<pre>public int countLineItems() { //logic to count }  public boolean add(LinItemVO itemToAdd){     ...//additional logic or checks     return lineItems.add(itemToAdd); }  public boolean remove(LinItemVO itemToAdd){     return lineItems.remove(itemToAdd); }  public ListIterator getIterator(){ return lineItems.iterator();}</pre>
<p>Above approaches are bad because disparate objects are stored in the <b>lineItem</b> collection in example-1 and example-2 relies on indices to store disparate objects. The indices based approach and storing disparate objects are hard to maintain and understand because indices are hard coded and get scattered across the code. If an index position changes for some reason, then you will have to change every occurrence, otherwise it breaks your application.</p> <p>The above coding approaches are analogous to storing disparate items in a storage system without proper labelling and just relying on its grid position.</p>	<p>Now to use:</p> <pre>Order myOrder = new OverseasOrder(1234) ;  LinItemVO item1 = new LinItemVO(); item1.setItemId(1); item1.setProductName("BBQ");  LinItemVO item2 = new LinItemVO(); item2.setItemId(2); item2.setProductName("Outdoor chair");  //to add line items to order myOrder.add(item1); myOrder.add(item2); ...</pre>

**Q 16:** When providing a user defined key class for storing objects in the Hashmaps or Hashtables, what methods do you have to provide or override (i.e. **method overriding**)? **LF** **P** **CO**

**A 16:** You should override the **equals()** and **hashCode()** methods from the *Object* class. The default implementation of the **equals()** and **hashcode()**, which are inherited from the *java.lang.Object* uses an object instance's memory location (e.g. *MyObject@6c60f2ea*). This can cause problems when two instances of the car objects have the same colour but the inherited **equals()** will return false because it uses the memory location, which is different for the two instances. Also the **toString()** method can be overridden to provide a proper string representation of your object. **Points to consider:**

- If a class overrides **equals()**, it must override **hashCode()**.
- If 2 objects are equal, then their hashCode values must be equal as well.
- If a field is not used in **equals()**, then it must not be used in **hashCode()**.
- If it is accessed often, **hashCode()** is a candidate for caching to enhance performance.

**Note:** Java 1.5 introduces enumerated constants, which improves readability and maintainability of your code. Java programming language enums are more powerful than their counterparts in other languages. E.g. A class like *Weather* can be built on top of simple enum type *Season* and the class *Weather* can be made immutable, and only one instance of each *Weather* can be created, so that your *Weather* class **does not have to override equals() and hashCode() methods**.

```
public class Weather {
 public enum Season {WINTER, SPRING, SUMMER, FALL}
 private final Season season;
 private static final List<Weather> listWeather = new ArrayList<Weather>();

 private Weather (Season season) { this.season = season;}
 public Season getSeason () { return season;}

 static {
 for (Season season : Season.values()) {
 listWeather.add(new Weather(season));
 }
 }

 public static ArrayList<Weather> getWeatherList () { return listWeather; }
 public String toString(){ return season;} // takes advantage of toString() method of Season.
}
```

**Q 17:** What is the main difference between a String and a StringBuffer class? **LF PI CI CO**

**A 17:**

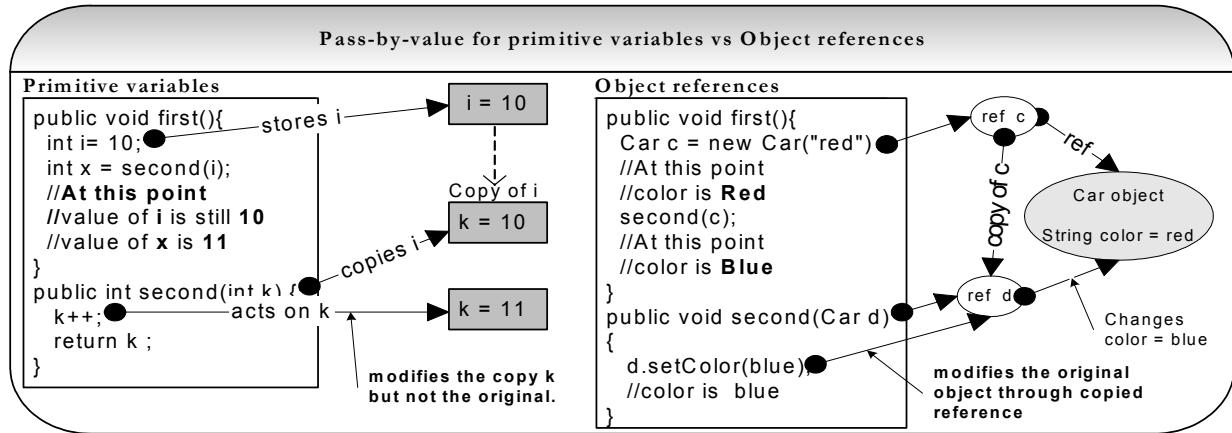
String	StringBuffer / StringBuilder
<p><b>String is immutable:</b> you can't modify a string object but can replace it by creating a new instance. Creating a new instance is rather expensive.</p> <pre>//Inefficient version using immutable String String output = "Some text" Int count = 100; for(int i=0; i&lt;count; i++) {     output += i; } return output;</pre> <p>The above code would build 99 new String objects, of which 98 would be thrown away immediately. Creating new objects is not efficient.</p>	<p><b>StringBuffer is mutable:</b> use StringBuffer or StringBuilder when you want to modify the contents. <b>StringBuilder</b> was added in Java 5 and it is identical in all respects to <b>StringBuffer</b> except that it is not synchronised, which makes it slightly faster at the cost of not being thread-safe.</p> <pre>//More efficient version using mutable StringBuffer StringBuffer output = new StringBuffer(110); Output.append("Some text"); for(int i=0; i&lt;count; i++) {     output.append(i); } return output.toString();</pre> <p>The above code creates only two new objects, the <b>StringBuffer</b> and the final <b>String</b> that is returned. StringBuffer expands as needed, which is costly however, so it would be better to initialise the <b>StringBuffer</b> with the correct size from the start as shown.</p>

Another important point is that creation of extra strings is not limited to 'overloaded mathematical operators' ("+"") but there are several methods like **concat()**, **trim()**, **substring()**, and **replace()** in String classes that generate new string instances. So use StringBuffer or StringBuilder for computation intensive operations, which offer better performance.

**Q 18:** What is the main difference between pass-by-reference and pass-by-value? **LF PI**

**A 18:** Other languages use **pass-by-reference** or pass-by-pointer. But in Java no matter what type of argument you pass the corresponding parameter (primitive variable or object reference) will get a copy of that data, which is exactly how **pass-by-value** (i.e. copy-by-value) works.

In Java, if a calling method passes a reference of an object as an argument to the called method then the **passed-in reference gets copied first** and then passed to the called method. Both the original reference that was passed-in and the copied reference will be pointing to the same object. So no matter which reference you use, you will be always modifying the same original object, which is how the **pass-by-reference works as well**.

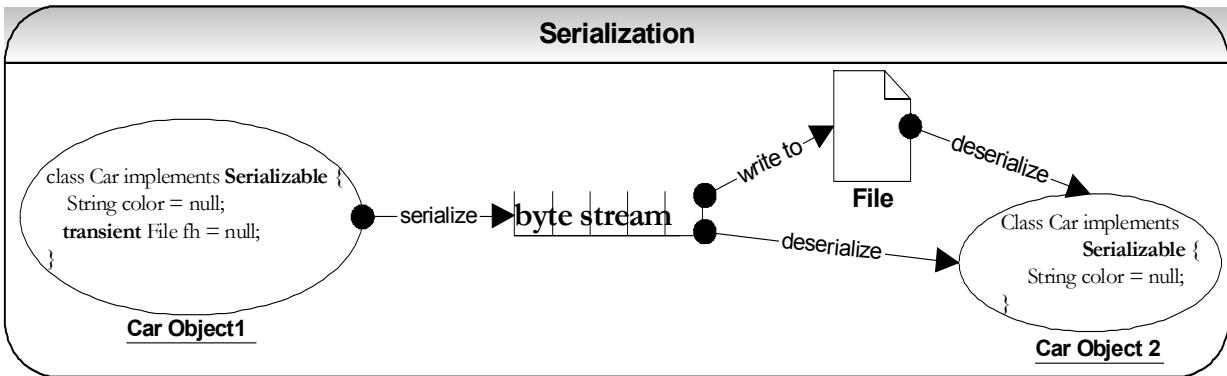


If your method call involves inter-process (e.g. between two JVMs) communication, then the reference of the calling method has a different address space to the called method sitting in a separate process (i.e. separate JVM). Hence inter-process communication involves calling method passing objects as arguments to called method by-value in a serialized form, which can adversely affect performance due to marshalling and unmarshalling cost.

**Note:** As discussed in Q69 in Enterprise section, EJB 2.x introduced local interfaces, where enterprise beans that can be used locally within the same JVM using Java's form of **pass-by-reference**, hence improving performance.

**Q 19:** What is serialization? How would you exclude a field of a class from serialization or what is a transient variable?  
What is the common use? **[LF S PI]**

**A 19:** Serialization is a process of reading or writing an object. It is a process of saving an object's state to a sequence of bytes, as well as a process of rebuilding those bytes back into a live object at some future time. An object is marked serializable by implementing the `java.io.Serializable` interface, which is only a *marker* interface -- it simply allows the serialization mechanism to verify that the class can be persisted, typically to a file.



**Transient** variables cannot be serialized. The fields marked **transient** in a serializable object will not be transmitted in the byte stream. An example would be a file handle or a database connection. Such objects are only meaningful locally. So they should be marked as transient in a serializable class.

Serialization can adversely affect performance since it:

- Depends on reflection.
- Has an incredibly verbose data format.
- Is very easy to send surplus data.

**When to use serialization?** Do not use serialization if you do not have to. A common use of serialization is to use it to send an object over the network or if the state of an object needs to be persisted to a flat file or a database. (Refer **Q57** on Enterprise section). Deep cloning or copy can be achieved through serialization. This may be fast to code but will have performance implications (Refer **Q22** in Java section).

The **objects stored in an HTTP session should be serializable** to support in-memory replication of sessions to achieve scalability (Refer **Q20** in Enterprise section). Objects are passed in RMI (Remote Method Invocation) across network using serialization (Refer **Q57** in Enterprise section).

**Q 20:** Explain the Java I/O streaming concept and the use of the decorator design pattern in Java I/O? **[LF DP PI SI]**

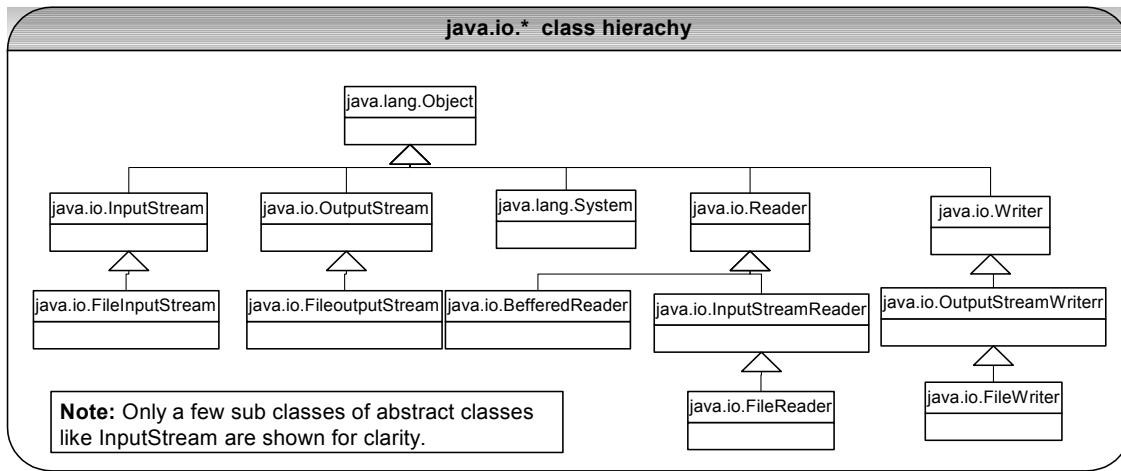
**A 20:** Java input and output is defined in terms of an abstract concept called a “**stream**”, which is a sequence of data. There are 2 kinds of streams.

- Byte streams (8 bit bytes) → Abstract classes are: **InputStream** and **OutputStream**
- Character streams (16 bit UNICODE) → Abstract classes are: **Reader** and **Writer**

**Design pattern:** `java.io.*` classes use the **decorator design pattern**. The decorator design pattern **attaches responsibilities to objects at runtime**. Decorators are more flexible than inheritance because the **inheritance attaches responsibility to classes at compile time**. The `java.io.*` classes use the decorator pattern to construct different combinations of behaviour at runtime based on some basic classes.

Attaching responsibilities to classes at compile time using subclassing.	Attaching responsibilities to objects at runtime using a decorator design pattern.
Inheritance (aka subclassing) attaches responsibilities to classes at compile time. When you extend a class, each individual changes you make to child class will affect all instances of the child classes. Defining many classes using inheritance to have all possible combinations is problematic and inflexible.	<p>By attaching responsibilities to <b>objects at runtime</b>, you can apply changes to each individual object you want to change.</p> <pre>File file = new File("c:/temp"); FileInputStream fis = new FileInputStream(file); BufferedInputStream bis = new BufferedInputStream(fis);</pre> <p>Decorators decorate an object by enhancing or restricting functionality of an object it decorates. The decorators add or restrict functionality to decorated</p>

objects either before or after forwarding the request. At runtime the `BufferedInputStream` (bis), which is a **decorator** (aka a **wrapper** around decorated object), forwards the method call to its **decorated** object `InputStream` (fis). The 'bis' will apply the additional functionality of buffering around the lower level file (i.e. fis) I/O.



### The New I/O (NIO): more scalable and better performance

Java has long been not suited for developing programs that perform a lot of I/O operations. Furthermore, commonly needed tasks such as file locking, non-blocking and asynchronous I/O operations and ability to map file to memory were not available. Non-blocking I/O operations were achieved through work around such as multithreading or using JNI. The **New I/O API** (aka **NIO**) in J2SE 1.4 has changed this situation.

A server's ability to handle several client requests effectively depends on how it uses I/O streams. When a server has to handle hundreds of clients simultaneously, it must be able to use I/O services concurrently. One way to cater for this scenario in Java is to use threads but having almost one-to-one ratio of threads (100 clients will have 100 threads) is prone to enormous **thread overhead and can result in performance and scalability problems due to consumption of memory stacks and CPU context switching**. To overcome this problem, a new set of non-blocking I/O classes have been introduced to the Java platform in `java.nio` package. The non-blocking I/O mechanism is built around **Selectors** and **Channels**. **Channels**, **Buffers** and **Selectors** are the core of the NIO.

A **Channel** class represents a bi-directional communication channel (similar to `InputStream` and `OutputStream`) between datasources such as a socket, a file, or an application component, which is capable of performing one or more I/O operations such as reading or writing. Channels can be non-blocking, which means, no I/O operation will wait for data to be read or written to the network. The good thing about NIO channels is that they can be asynchronously interrupted and closed. So if a thread is blocked in an I/O operation on a channel, another thread can interrupt that blocked thread.

**Buffers** hold data. Channels can fill and drain **Buffers**. Buffers replace the need for you to do your own buffer management using byte arrays. There are different types of Buffers like `ByteBuffer`, `CharBuffer`, `DoubleBuffer`, etc.

A **Selector** class is responsible for multiplexing (combining multiple streams into a single stream) by allowing a single thread to service multiple channels. Each **Channel** registers events with a **Selector**. When events arrive from clients, the Selector demultiplexes (separating a single stream into multiple streams) them and dispatches the events to corresponding Channels. To achieve non-blocking I/O a **Channel** class must work in conjunction with a **Selector** class.

**Design pattern:** NIO uses a **reactor design pattern**, which demultiplexes events (separating single stream into multiple streams) and dispatches them to registered object handlers. The reactor pattern is similar to an **observer pattern** (aka publisher and subscriber design pattern), but an observer pattern handles only a single source of events (i.e. a single publisher with multiple subscribers) where a reactor pattern handles multiple event sources (i.e. multiple publishers with multiple subscribers). The intent of an observer pattern is to define a one-to-many dependency so that when one object (i.e. the publisher) changes its state, all its dependents (i.e. all its subscribers) are notified and updated correspondingly.

Another sought after functionality of NIO is its ability to map a file to memory. There is a specialized form of a Buffer known as `MappedByteBuffer`, which represents a buffer of bytes mapped to a file. To map a file to

MappedByteBuffer, you must first get a channel for a file. Once you get a channel then you map it to a buffer and subsequently you can access it like any other ByteBuffer. Once you map an input file to a CharBuffer, you can do pattern matching on the file contents. This is similar to running "grep" on a UNIX file system.

Another feature of NIO is its ability to lock and unlock files. Locks can be exclusive or shared and can be held on a contiguous portion of a file. But file locks are subject to the control of the underlying operating system.

**Q 21:** How can you improve Java I/O performance? **PI** **BP**

**A 21:** Java applications that utilise Input/Output are excellent candidates for performance tuning. Profiling of Java applications that handle significant volumes of data will show significant time spent in I/O operations. This means substantial gains can be had from I/O performance tuning. Therefore, I/O efficiency should be a high priority for developers looking to optimally increase performance.

The basic rules for speeding up I/O performance are

- Minimise accessing the hard disk.
- Minimise accessing the underlying operating system.
- Minimise processing bytes and characters individually.

Let us look at some of the techniques to improve I/O performance. **CO**

- Use **buffering** to minimise disk access and underlying operating system. As shown below, with buffering large chunks of a file are read from a disk and then accessed a byte or character at a time.

Without buffering : inefficient code	With Buffering: yields better performance
<pre>try{     File f = new File("myFile.txt");     FileInputStream fis = new FileInputStream(f);     int count = 0;     int b = ;     while((b = fis.read()) != -1){         if(b== '\n'){             count++;         }     }     // fis should be closed in a finally block.     fis.close(); } catch(IOException io){} <b>Note:</b> fis.read() is a native method call to the underlying system.</pre>	<pre>try{     File f = new File("myFile.txt");     FileInputStream fis = new FileInputStream(f);     BufferedInputStream bis = new BufferedInputStream(fis);     int count = 0;     int b = ;     while((b = bis.read()) != -1){         if(b== '\n'){             count++;         }     }     //bis should be closed in a finally block.     bis.close(); } catch(IOException io){} <b>Note:</b> bis.read() takes the next byte from the input buffer and only rarely access the underlying operating system.</pre>

Instead of reading a character or a byte at a time, the above code with buffering can be improved further by reading one line at a time as shown below:

```
FileReader fr = new FileReader(f);
BufferedReader br = new BufferedReader(fr);
While (br.readLine() != null) count++;
```

By default the **System.out** is line buffered, which means that the output buffer is flushed when a new line character is encountered. This is required for any interactivity between an input prompt and display of output. The line buffering can be disabled for faster I/O operation as follows:

```
FileOutputStream fos = new FileOutputStream(file);
BufferedOutputStream bos = new BufferedOutputStream(fos, 1024);
PrintStream ps = new PrintStream(bos,false);
System.setOut(ps);

while (someConditionIsTrue)
 System.out.println("blah...blah...");
```

It is recommended to use logging frameworks like **Log4J** or **apache commons logging**, which uses buffering instead of using default behaviour of **System.out.println(....)** for better performance. Frameworks like Log4J are configurable, flexible, extensible and easy to use.

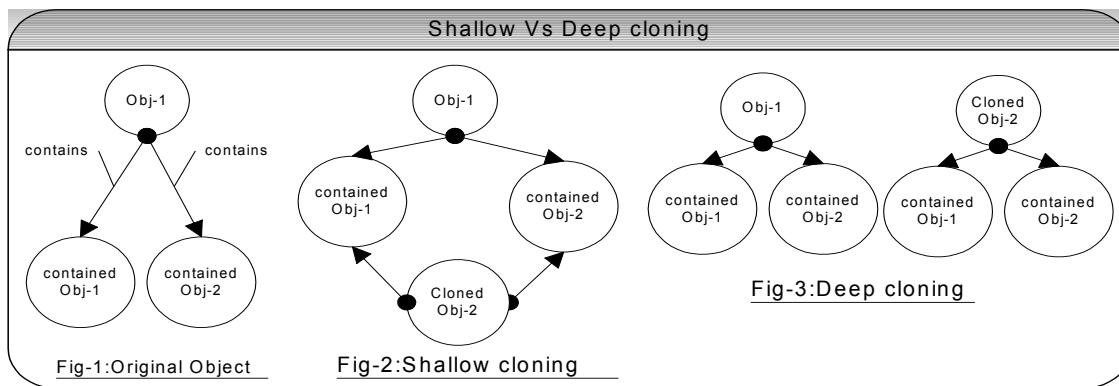
- Use the NIO package, if you are using JDK 1.4 or later, which uses performance-enhancing features like buffers to hold data, memory mapping of files, non-blocking I/O operations etc.
- I/O performance can be improved by minimising the calls to the underlying operating systems. The Java runtime itself cannot know the length of a file, querying the file system for isDirectory(), isFile(), exists() etc must query the underlying operating system.
- Where applicable caching can be used to improve performance by reading in all the lines of a file into a Java collection class like an ArrayList or a HashMap and subsequently access the data from an in-memory collection instead of the disk.

**Q 22:** What is the main difference between shallow cloning and deep cloning of objects? **DC LF MI PI**

**A 22:** The default behaviour of an object's clone() method automatically yields a shallow copy. So to achieve a deep copy the classes must be edited or adjusted.

**Shallow copy:** If a shallow copy is performed on obj-1 as shown in fig-2 then it is copied but its contained objects are not. The contained objects Obj-1 and Obj-2 are affected by changes to cloned Obj-2. Java supports shallow cloning of objects by default when a class implements the `java.lang.Cloneable` interface.

**Deep copy:** If a deep copy is performed on obj-1 as shown in fig-3 then not only obj-1 has been copied but the objects contained within it have been copied as well. Serialization can be used to achieve deep cloning. Deep cloning through serialization is faster to develop and easier to maintain but carries a performance overhead.



**For example**, invoking `clone()` method on a `HashMap` returns a shallow copy of `HashMap` instance, which means **the keys and values themselves are not cloned**. If you want a deep copy then a simple method is to serialize the `HashMap` to a `ByteArrayOutputStream` and then deserialize it. This creates a deep copy but does require that all keys and values in the `HashMap` are `Serializable`. Its primary advantage is that it will deep copy any arbitrary object graph.

**List some of the methods supported by Java object class?** `clone()`, `toString()`, `equals(Object obj)`, `hashCode()` → refer **Q16** in Java section, `wait()`, `notify()` → refer **Q42** in Java section, `finalize()` etc.

**Q 23:** What is the difference between an instance variable and a static variable? Give an example where you might use a static variable? **LF**

**A 23:**

Static variable	Instance variable
Class variables are called static variables. There is only one occurrence of a class variable per JVM per class loader. When a class is loaded the class variables (aka static variables) are initialised.	Instance variables are non-static and there is one occurrence of an instance variable in each class instance (i.e. each object).

A static variable is used in the **singleton** pattern. (Refer **Q45** in Java section). A static variable is used with a **final** modifier to define **constants**.

**Q 24:** Give an example where you might use a static method? **LF**

**A 24:** Static methods prove useful for creating **utility classes**, **singleton classes** and **factory methods** (Refer Q45, Q46 in Java section). Utility classes are not meant to be instantiated. Improper coding of utility classes can lead to procedural coding. **java.lang.Math**, **java.util.Collections** etc are examples of utility classes in Java.

**Q 25:** What are access modifiers? **LF**

**A 25:**

Modifier	Used with	Description
public	Outer classes, interfaces, constructors, Inner classes, methods and field variables	A class or interface may be accessed from outside the package. Constructors, inner classes, methods and field variables <b>may be accessed wherever their class is accessed</b> .
protected	Constructors, inner classes, methods, and field variables.	Accessed by other classes in the <b>same package or any subclasses of the class</b> in which they are referred (i.e. <b>same package or different package</b> ).
private	Constructors, inner classes, methods and field variables,	Accessed only <b>within the class in which they are declared</b>
No modifier: (Package by default).	Outer classes, inner classes, interfaces, constructors, methods, and field variables	Accessed only <b>from within the package in which they are declared</b> .

**Q 26:** Where and how can you use a private constructor? **LF**

**A 26:** Private constructor is used if you do not want other classes to instantiate the object. The instantiation is done by a public static method within the same class.

- Used in the singleton pattern. (Refer Q45 in Java section).
- Used in the factory method pattern (Refer Q46 in Java section).
- Used in utility classes e.g. StringUtils etc.

**Q 27:** What is a final modifier? Explain other Java modifiers? **LF**

**A 27:** A final class can't be extended i.e. A final class may not be subclassed. A final method can't be overridden when its class is inherited. You can't change value of a final variable (i.e. it is a constant).

Modifier	Class	Method	Property
static	A static inner class is just an inner class associated with the class, rather than with an instance.	cannot be instantiated, are called by classname.method, can only access static variables	Only one instance of the variable exists.
abstract	Cannot be instantiated, must be a superclass, used whenever one or more methods are abstract.	Method is defined but contains no implementation code (implementation code is included in the subclass). If a method is abstract then the entire class must be abstract.	N/A
synchronized	N/A	Acquires a <b>lock on the class for static methods</b> . Acquires a <b>lock on the instance for non-static methods</b> .	N/A
transient	N/A	N/A	Field should not be serialized.
final	Class cannot be inherited	Method cannot be overridden	Makes the variable a constant.
native	N/A	Platform dependent. No body, only signature.	N/A

**Note:** Be prepared for tricky questions on modifiers like, what is a “**volatile**”? Or what is a “**const**”? Etc. The reason it is tricky is that Java does have these keywords “**const**” and “**volatile**” as reserved, which means you can't name your variables with these names **but modifier “const” is not yet added in the language** and the **modifier “volatile” is very rarely used**.

The “**volatile**” modifier is used on member variables that may be modified simultaneously by other threads. Since other threads cannot see local variables, there is no need to mark local variables as volatile. E.g. **volatile int** number; **volatile private List** listItems = null; etc. The modifier **volatile** only synchronizes the variable marked as **volatile** whereas “**synchronized**” modifier synchronizes all variables.

Java uses the final modifier to declare constants. A final variable or constant declared as “**final**” has a value that is immutable and cannot be modified to refer to any other objects other than one it was initialized to refer to. So the “**final**” modifier applies only to the value of the variable itself, and not to the object referenced by the variable. This is where the “**const**” modifier can come in **very useful if added to the Java language**. A reference variable or a constant marked as “**const**” refers to an immutable object that cannot be modified. The reference variable itself can be modified, if it is not marked as “**final**”. The “**const**” modifier will be applicable only to non-primitive types. The primitive types should continue to use the modifier “**final**”.

**Q 28:** What is the difference between final, finally and finalize() in Java? **[LF]**

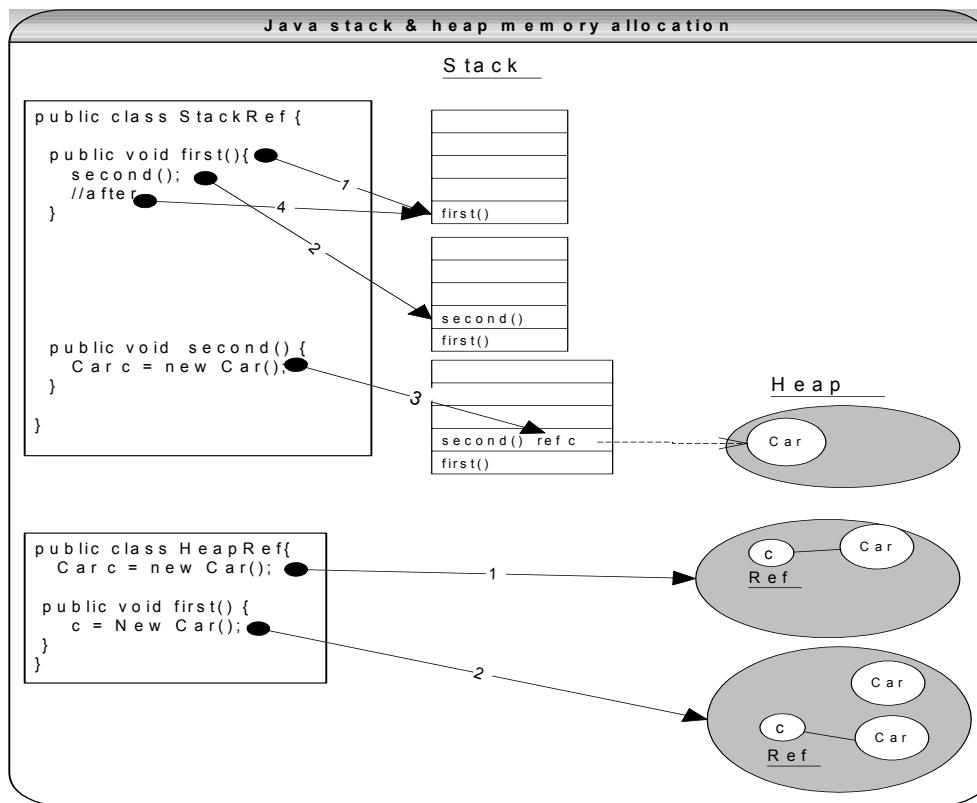
**A 28:**

- **final** - constant declaration. Refer **Q27** in Java section.
- **finally** - handles exception. The finally block is optional and provides a mechanism to clean up regardless of what happens within the try block (except System.exit(0) call). Use the finally block to close files or to release other system resources like database connections, statements etc. (Refer **Q45** in Enterprise section)
- **finalize()** - method helps in garbage collection. A **method** that is invoked before an object is discarded by the garbage collector, allowing it to clean up its state. Should not be used to release non-memory resources like file handles, sockets, database connections etc because Java has only a finite number of these resources and you do not know when the garbage collection is going to kick in to release these non-memory resources through the finalize() method.

**Q 29:** How does Java allocate stack and heap memory? Explain re-entrant, recursive and idempotent methods/functions? **[MC]**

**A 29:** Each time an object is created in Java it goes into the area of memory known as **heap**. The primitive variables like int and double are allocated in the **stack**, if they are local method variables and in the **heap** if they are member variables (i.e. fields of a class). In Java methods local variables are pushed into stack when a method is invoked and stack pointer is decremented when a method call is completed. In a multi-threaded application each thread will have its own stack but will share the same heap. This is why care should be taken in your code to avoid any concurrent access issues in the heap space. The stack is threadsafe (each thread will have its own stack) but the heap is not threadsafe unless guarded with synchronisation through your code.

A method in stack is **re-entrant** allowing multiple concurrent invocations that do not interfere with each other. A function is **recursive** if it calls itself. Given enough stack space, recursive method calls are perfectly valid in Java though it is tough to debug. Recursive functions are useful in removing iterations from many sorts of algorithms. All recursive functions are re-entrant but not all re-entrant functions are recursive. **Idempotent** methods are methods, which are written in such a way that repeated calls to the same method with the same arguments yield same results. For example clustered EJBs, which are written with idempotent methods, can automatically recover from a server failure as long as it can reach another server.



**Q 30:** Explain Outer and Inner classes (or Nested classes) in Java? When will you use an Inner Class? **[LF]**

**A 30:** In Java not all classes have to be defined separate from each other. You can put the definition of one class inside the definition of another class. The inside class is called an inner class and the enclosing class is called an outer class. So when you define an inner class, it is a member of the outer class in much the same way as other members like attributes, methods and constructors.

**Where should you use inner classes?** Code **without** inner classes is **more maintainable** and **readable**. When you access private data members of the outer class, the JDK compiler creates package-access member functions in the outer class for the inner class to access the private members. This leaves a **security hole**. In general **we should avoid using inner classes**. Use inner class only when an inner class is only relevant in the context of the outer class and/or inner class can be made private so that only outer class can access it. Inner classes are used primarily to implement helper classes like Iterators, Comparators etc which are used in the context of an outer class. **CO**

Member inner class	Anonymous inner class
<pre>public class MyStack {     private Object[] items = null;     ...     public Iterator iterator() {         return new StackIterator();     }     //inner class     class StackIterator implements Iterator{         ...         public boolean hasNext(){...}     } }</pre>	<pre>public class MyStack {     private Object[] items = null;     ...     public Iterator iterator() {         return new Iterator {             ...             public boolean hasNext() {...}         }     } }</pre>

#### Explain outer and inner classes?

Class Type	Description	Example + Class name
Outer class	Package member class or interface	//package scope <b>class Outside{}</b>  Outside.class
Inner class	static nested class or interface	//package scope class Outside { static class Inside{} }  Outside.class , Outside\$Inside.class
Inner class	Member class	class Outside{ class Inside{} }  Outside.class , Outside\$Inside.class
Inner class	Local class	class Outside { void first() { final int i = 5; class Inside{} } }  Outside.class , Outside\$1\$Inside.class
Inner class	Anonymous class	class Outside{ void first() { button.addActionListener ( new ActionListener() { public void actionPerformed(ActionEvent e) { System.out.println("The button was pressed!"); } }); } }  Outside.class , Outside\$1.class

**Q 31:** What is type casting? Explain up casting vs. down casting? When do you get ClassCastException? **LF DP**

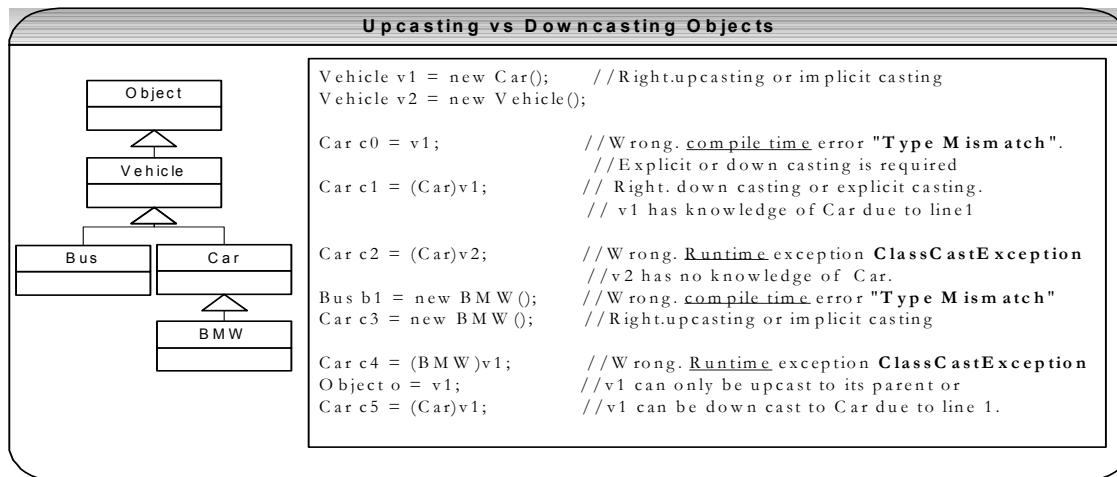
**A 31:** Type casting means treating a variable of one type as though it is another type.

When up casting **primitives** as shown below from left to right, automatic conversion occurs. But if you go from right to left, down casting or explicit casting is required. Casting in Java is safer than in C or other languages that allow arbitrary casting. Java only lets casts occur when they make sense, such as a cast between a float and an int. However you can't cast between an int and a *String* (is an object in Java).

**byte → short → int → long → float → double**

```
int i = 5;
long j = i; //Right. Up casting or implicit casting
byte b1 = i; //Wrong. Compile time error "Type Mismatch".
byte b2 = (byte) i; //Right. Down casting or explicit casting is required.
```

When it comes to object references you can always cast from a subclass to a superclass because a subclass object is also a superclass object. You can cast an object implicitly to a super class type (i.e. **upcasting**). If this were not the case **polymorphism wouldn't be possible**.



You can cast down the hierarchy as well but you must explicitly write the cast and the **object must be a legitimate instance of the class you are casting to**. The **ClassCastException** is thrown to indicate that code has attempted to cast an object to a subclass of which it is not an instance. We can deal with the problem of incorrect casting in two ways:

- Use the exception handling mechanism to catch **ClassCastException**.

```

try{
 Object o = new Integer(1);
 System.out.println((String)o);
}
catch(ClassCastExceptioncce){
 logger.log("Invalid casting, String is expected...Not an Integer");
 System.out.println(((Integer)o).toString());
}

```

- Use the **instanceof** statement to guard against incorrect casting.

```

if(v2 instanceof Car){
 Car c2 = (Car)v2;
}

```

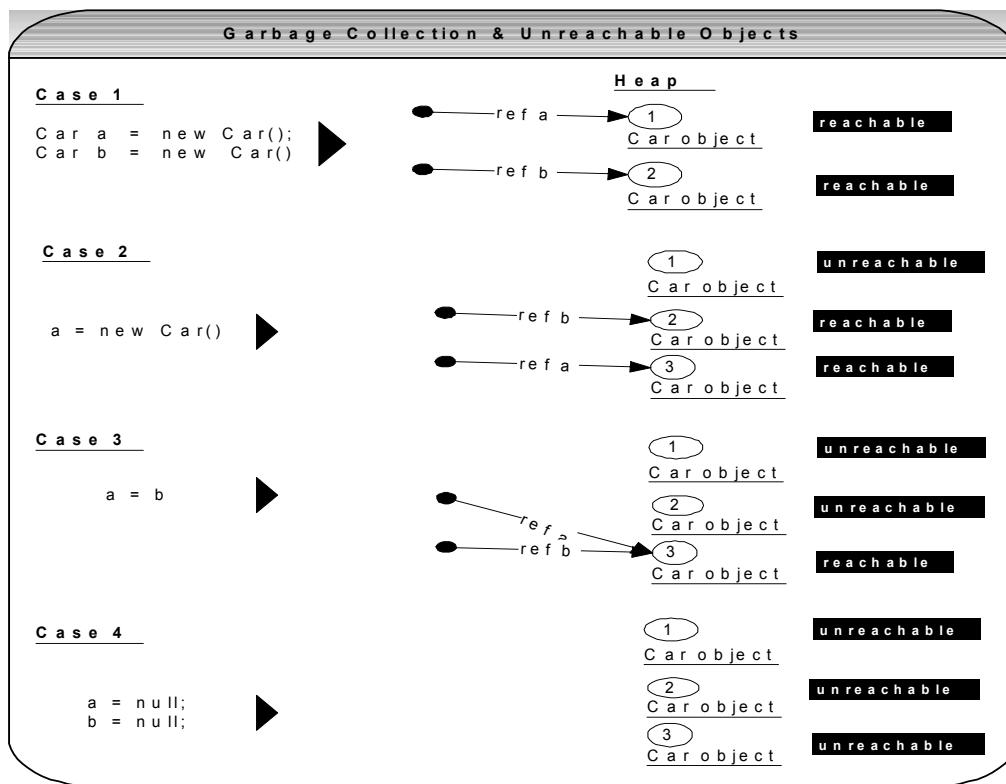
**Design pattern:** The “**instanceof**” and “**typecast**” constructs are shown for the illustration purpose only. Using these constructs can be unmaintainable due to large if and elseif statements and can affect performance if used in frequently accessed methods or loops. Look at using **visitor design pattern** to avoid these constructs. (Refer **Q11** in How would you go about section...).

**Points-to-ponder:** We can also get a **ClassCastException** when two different class loaders load the same class because they are treated as two different classes.

**Q 32:** What do you know about the Java garbage collector? When does the garbage collection occur? Explain different types of references in Java? **[LF MI]**

**A 32:** Each time an object is created in Java, it goes into the area of memory known as heap. The Java heap is called the garbage collectable heap. The garbage collection **cannot be forced**. The garbage collector runs in low memory situations. When it runs, it releases the memory allocated by an unreachable object. The garbage collector runs on a low priority daemon (background) thread. You can **nicely ask** the garbage collector to collect garbage by calling `System.gc()` but **you can't force it**.

**What is an unreachable object?** An object's life has no meaning unless something has reference to it. If you can't reach it then you can't ask it to do anything. Then the object becomes unreachable and the garbage collector will figure it out. Java automatically collects all the unreachable objects periodically and releases the memory consumed by those unreachable objects to be used by the future reachable objects.



We can use the following options with the **Java** command to enable tracing for garbage collection events.

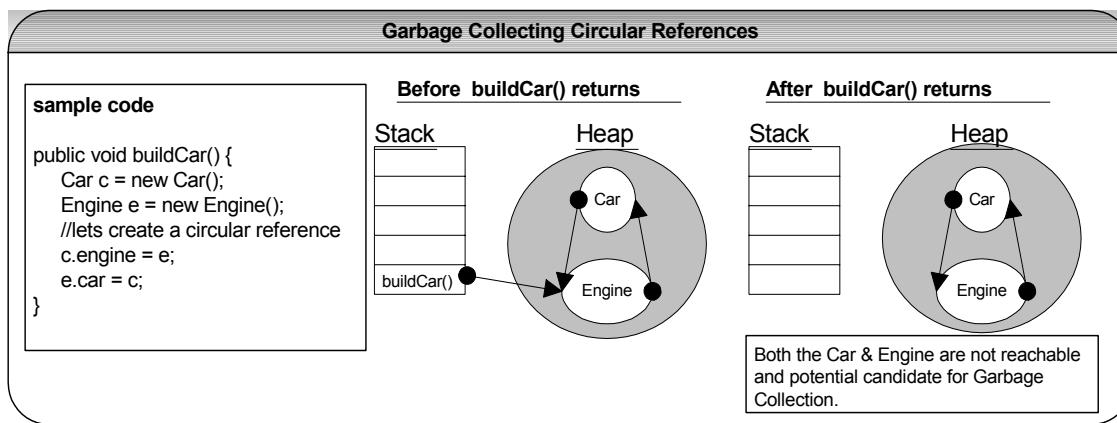
**-verbose:gc** reports on each garbage collection event.

**Explain types of references in Java?** `java.lang.ref` package can be used to declare soft, weak and phantom references.

- Garbage Collector won't remove a **strong reference**.
- A **soft reference** **will** only get removed if memory is low. So it is useful for implementing caches while avoiding memory leaks.
- A **weak reference** will get removed on the next garbage collection cycle. Can be used for implementing canonical maps. The `java.util.WeakHashMap` implements a `HashMap` with keys held by weak references.
- A **phantom reference** will be finalized but the memory will not be reclaimed. Can be useful when you want to be notified that an object is about to be collected.

**Q 33:** If you have a circular reference of objects, but you no longer reference it from an execution thread, will this object be a potential candidate for garbage collection? **[LF MI]**

**A 33:** Yes. Refer diagram below.

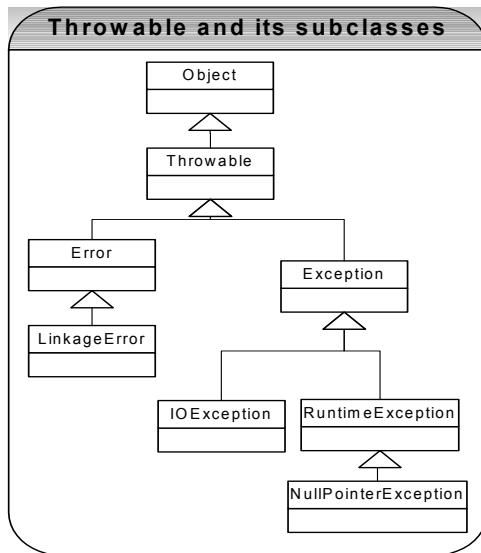


**Q 34:** Discuss the Java error handling mechanism? What is the difference between Runtime (**unchecked**) exceptions and **checked** exceptions? What is the implication of catching all the exceptions with the type “*Exception*”? **EH** **BP**

**A 34:**

**Errors:** When a dynamic linking failure or some other “hard” failure in the virtual machine occurs, the virtual machine throws an Error. Typical Java programs should not catch Errors. In addition, it’s unlikely that typical Java programs will ever throw Errors either.

**Exceptions:** Most programs throw and catch objects that derive from the Exception class. Exceptions indicate that a problem occurred but that the problem is not a serious JVM problem. An Exception class has many subclasses. These descendants indicate various types of exceptions that can occur. For example, NegativeArraySizeException indicates that a program attempted to create an array with a negative size. One exception subclass has special meaning in the Java language: RuntimeException. All the exceptions except RuntimeException are compiler checked exceptions. If a method is capable of throwing a checked exception it must declare it in its method header or handle it in a try/catch block. Failure to do so raises a compiler error. So checked exceptions can, at compile time, greatly reduce the occurrence of unhandled exceptions surfacing at runtime in a given application at the expense of requiring large throws declarations and encouraging use of poorly-constructed try/catch blocks. Checked exceptions are present in other languages like C++, C#, and Python.



#### ***Runtime Exceptions (unchecked exception)***

A RuntimeException class represents exceptions that occur within the Java virtual machine (during runtime). An example of a runtime exception is NullPointerException. The cost of checking for the runtime exception often outweighs the benefit of catching it. Attempting to catch or specify all of them all the time would make your code unreadable and unmaintainable. The compiler allows runtime exceptions to go uncaught and unspecified. If you

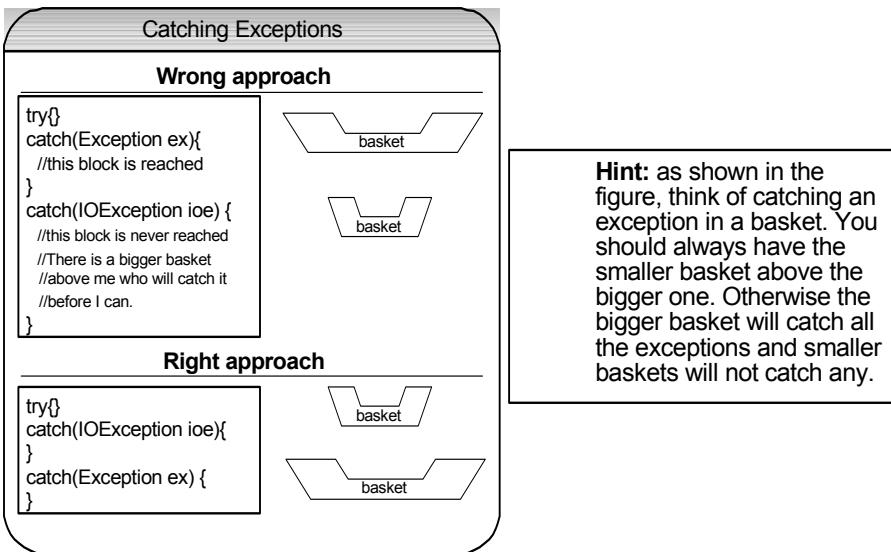
like, you can catch these exceptions just like other exceptions. However, you do not have to declare it in your "throws" clause or catch it in your catch clause. In addition, you can create your own *RuntimeException* subclasses and this approach is probably preferred at times because checked exceptions can complicate method signatures and can be difficult to follow.

### **Exception handling best practices:** BP

#### **Why is it not advisable to catch type “Exception”?** CO

Exception handling in Java is **polymorphic** in nature. For example if you catch type *Exception* in your code then it can catch or throw its descendent types like *IOException* as well. So if you catch the type *Exception* before the type *IOException* then the type *Exception* block will catch the entire exceptions and type *IOException* block is never reached. In order to catch the type *IOException* and handle it differently to type *Exception*, *IOException* should be caught first (remember that you can't have a bigger basket above a smaller basket).

The diagram below is an example for illustration only. In practice it is not recommended to catch type “**Exception**”. We should only catch specific subtypes of the *Exception* class. Having a bigger basket (i.e. *Exception*) will hide or cause problems. Since the *RunTimeException* is a subtype of *Exception*, catching the type *Exception* will catch all the run time exceptions (like NullpointerException, ArrayIndexOut-OfBounds-Exception) as well.



#### **Why should you throw an exception early?** CO

The exception stack trace helps you pinpoint where an exception occurred by showing us the exact sequence of method calls that lead to the exception. By throwing your exception early, the exception becomes more accurate and more specific. Avoid suppressing or ignoring exceptions. Also avoid using exceptions just to get a flow control.

**Instead of:**

```
...
InputStream in = new FileInputStream(fileName); // assume this line throws an exception because filename == null.
...
```

**Use the following code because you get a more accurate stack trace:**

```
...
if(filename == null){
 throw new IllegalArgumentException("file name is null");
}

InputStream in = new FileInputStream(fileName);
...
```

#### **Why should you catch a checked exception late in a catch {} block?**

You should not try to catch the exception before your program can handle it in an appropriate manner. The natural tendency when a compiler complains about a checked exception is to catch it so that the compiler stops reporting

errors. The best practice is to catch the exception at the appropriate layer (e.g. an exception thrown at an integration layer can be caught at a presentation layer in a catch {} block), where your program can either meaningfully recover from the exception and continue to execute or log the exception only once in detail, so that user can identify the cause of the exception.

**Note:** Due to heavy use of checked exceptions and minimal use of unchecked exceptions, there has been a hot debate in the Java community regarding true value of checked exceptions. Use checked exceptions when the client code can take some useful recovery action based on information in exception. Use unchecked exception when client code cannot do anything. For example, convert your SQLException into another checked exception if the client code can recover from it and convert your SQLException into an unchecked (i.e. RuntimeException) exception, if the client code cannot do anything about it.

#### A note on key words for error handling:

**throw / throws** – used to pass an exception to the method that called it.

**try** – block of code will be tried but may cause an exception.

**catch** – declares the block of code, which handles the exception.

**finally** – block of code, which is always executed (except System.exit(0) call) no matter what program flow, occurs when dealing with an exception.

**assert** – Evaluates a conditional expression to verify the programmer's assumption.

**Q 35:** What is a user defined exception? **EH**

**A 35:** User defined exceptions may be implemented by defining a new exception class by extending the *Exception* class.

```
public class MyException extends Exception {

 /* class definition of constructors goes here */
 public MyException() {
 super();
 }

 public MyException (String errorMessage) {
 super (errorMessage);
 }
}
```

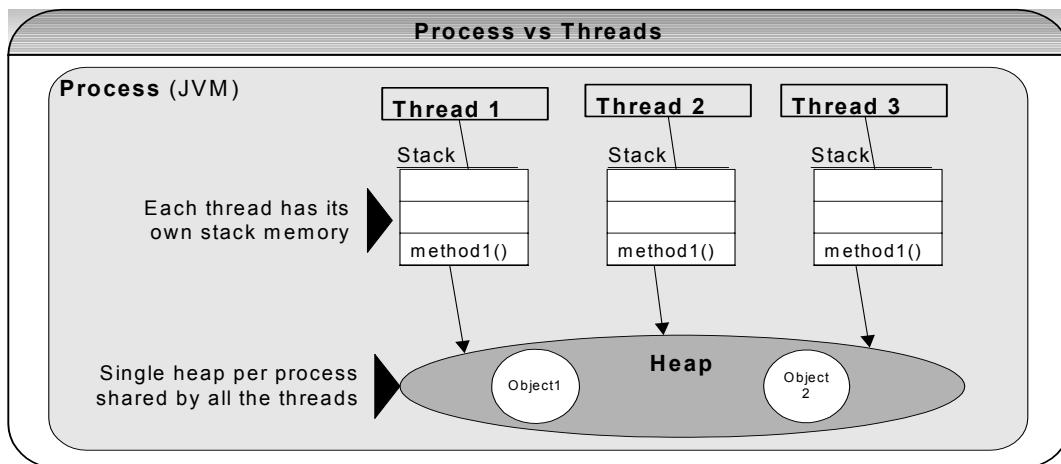
Throw and/or throws statement is used to signal the occurrence of an exception. Throw an exception:

**throw** new MyException("I threw my own exception.")

To declare an exception: **public myMethod() throws MyException {...}**

**Q 36:** What is the difference between processes and threads? **LF MI CI**

**A 36:** A process is an execution of a program but a thread is a single execution sequence within the process. A process can contain multiple threads. A thread is sometimes called a lightweight process.



A JVM runs in a single process and threads in a JVM share the heap belonging to that process. That is why several threads may access the same object. Threads **share the heap and have their own stack space**. This is

how one thread's invocation of a method and its local variables are kept thread safe from other threads. But the heap is not thread-safe and must be synchronized for thread safety.

**Q 37:** Explain different ways of creating a thread? **[LF]**

**A 37:** Threads can be used by either :

- Extending the **Thread** class
- Implementing the **Runnable** interface.

```
class Counter extends Thread {
 //method where the thread execution will start
 public void run(){
 //logic to execute in a thread
 }

 //let's see how to start the threads
 public static void main(String[] args){
 Thread t1 = new Counter();
 Thread t2 = new Counter();
 t1.start(); //start the first thread. This calls the run() method
 t2.start(); //this starts the 2nd thread. This calls the run() method
 }
}
```

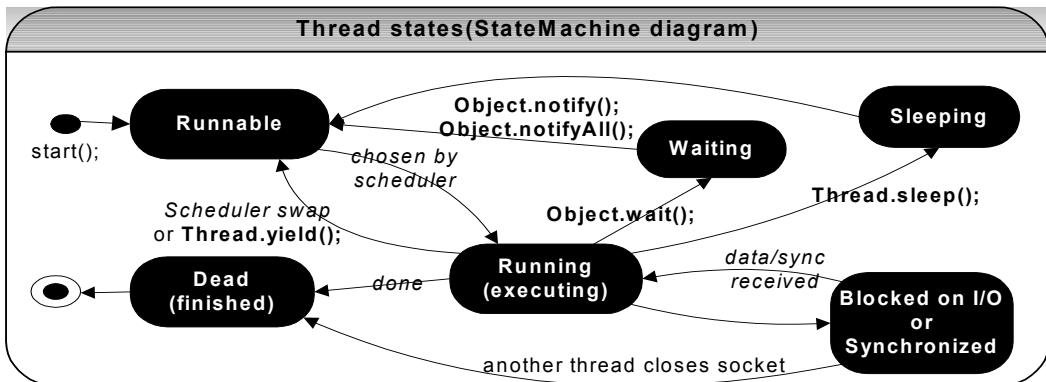
```
class Counter extends Base implements Runnable {
 //method where the thread execution will start
 public void run(){
 //logic to execute in a thread
 }

 //let us see how to start the threads
 public static void main(String[] args){
 Thread t1 = new Thread(new Counter());
 Thread t2 = new Thread(new Counter());
 t1.start(); //start the first thread. This calls the run() method
 t2.start(); //this starts the 2nd thread. This calls the run() method
 }
}
```

The runnable interface is preferred, as it does not require your object to inherit a thread because when you need multiple inheritance, only interfaces can help you. In the above example we had to extend the *Base* class so implementing runnable interface is an obvious choice. Also note how the threads are started in each of the different cases as shown in the code sample.

**Q 38:** Briefly explain high-level thread states? **[LF]**

**A 38:** The state chart diagram below describes the thread states. (Refer **Q107** in Enterprise section for state chart diagram).



(Diagram sourced from: <http://www.wilsonmar.com/1threads.htm>)

- **Runnable** — waiting for its turn to be picked for execution by the thread scheduler based on thread priorities.
- **Running**: The processor is actively executing the thread code. It runs until it becomes blocked, or voluntarily gives up its turn with this static method *Thread.yield()*. Because of context switching overhead, *yield()* should not be used very frequently.
- **Waiting**: A thread is in a **blocked state** while it waits for some external processing such as file I/O to finish.
- **Sleeping**: Java threads are forcibly put to sleep (suspended) with this overloaded method: *Thread.sleep(milliseconds)*, *Thread.sleep(milliseconds, nanoseconds)*;
- **Blocked on I/O**: Will move to runnable after I/O condition like reading bytes of data etc changes.
- **Blocked on synchronization**: Will move to Runnable when a **lock is acquired**.
- **Dead**: The thread is finished working.

**Q 39:** What is the difference between yield and sleeping? **[LF]**

**A 39:** When a task invokes *yield()*, it changes from running state to runnable state. When a task invokes *sleep()*, it changes from running state to waiting/sleeping state.

**Q 40:** How does thread synchronization occurs inside a monitor? What levels of synchronization can you apply? What is the difference between synchronized method and synchronized block? **[LF C PI]**

**A 40:** In Java programming, each object has a lock. A thread can acquire the lock for an object by using the **synchronized** keyword. The synchronized keyword can be applied in **method level** (coarse grained lock – can affect performance adversely) or **block level of code** (fine grained lock). Often using a lock on a method level is too coarse. Why lock up a piece of code that does not access any shared resources by locking up an entire method. Since each object has a lock, dummy objects can be created to implement block level synchronization. The block level is more efficient because it does not lock the whole method.

```
class MethodLevel {
 //shared among threads
 SharedResource x, y;

 public void synchronized
 method1() {
 //multiple threads can't access
 }

 public void synchronized
 method2() {
 //multiple threads can't access
 }

 public void method3() {
 //not synchronized
 //multiple threads can access
 }
}
```

```
class BlockLevel {
 //shared among threads
 SharedResource x, y;
 //dummy objects for locking
 Object xLock = new Object(), yLock = new Object();

 public void method1() {
 synchronized(xLock){
 //access x here. thread safe
 }
 //do something here but don't use
 SharedResource x, y;

 synchronized(xLock) {
 synchronized(yLock) {
 //access x,y here. thread safe
 }
 }
 //do something here but don't use
 SharedResource x, y;
 }
}
```

The JVM uses locks in conjunction with monitors. A monitor is basically a guardian who watches over a sequence of synchronized code and making sure only one thread at a time executes a synchronized piece of code. Each monitor is associated with an object reference. When a thread arrives at the first instruction in a block of code it must obtain a lock on the referenced object. The thread is not allowed to execute the code until it obtains the lock.

Once it has obtained the lock, the thread enters the block of protected code. When the thread leaves the block, no matter how it leaves the block, it releases the lock on the associated object.

**Why synchronization is important?** Without synchronization, it is possible for one thread to modify a shared object while another thread is in the process of using or updating that object's value. This often causes dirty data and leads to significant errors. **The disadvantage of synchronization** is that it can cause deadlocks when two threads are waiting on each other to do something. Also synchronized code has the overhead of acquiring lock, which can adversely affect the performance. **Deadlock in java is a part of multithreading. Deadlock can occur in a situation when a thread is waiting for an object lock, that is acquired by another thread and second thread is waiting for an object lock that is acquired by first thread. Since, both threads are waiting for each other to release the lock, the condition is called deadlock.**

**Q 41:** What is a daemon thread? **LF**

**A 41:** Daemon threads are sometimes called "service" threads. These are threads that normally run at a low priority and provide a basic service to a program or programs when activity on a machine is reduced. An example of a daemon thread that is continuously running is the garbage collector thread. This thread is provided by the JVM.

**Q 42:** How can threads communicate with each other? How would you implement a producer (one thread) and a consumer (another thread) passing data (via stack)? **LF**

**A 42:** The **wait()**, **notify()**, and **notifyAll()** methods are used to provide an efficient way for threads to communicate with each other. This communication solves the '**consumer-producer problem**'. This problem occurs when the producer thread is completing work that the other thread (consumer thread) will use.

**Example:** If you imagine an application in which one thread (the producer) writes data to a file while a second thread (the consumer) reads data from the same file. In this example the concurrent threads share the same resource file. Because these threads share the common resource file they should be synchronized. Also these two threads should communicate with each other because the consumer thread, which reads the file, should wait until the producer thread, which writes data to the file and notifies the consumer thread that it has completed its writing operation.

Let's look at a sample code where **count** is a shared resource. The consumer thread will wait inside the **consume()** method on the producer thread, until the producer thread increments the count inside the **produce()** method and subsequently notifies the consumer thread. Once it has been notified, the consumer thread waiting inside the **consume()** method will give up its waiting state and completes its method by consuming the count (i.e. decrementing the count).

**Thread communication (Consumer vs Producer threads)**

```

Class ConsumerProducer {
 private int count;

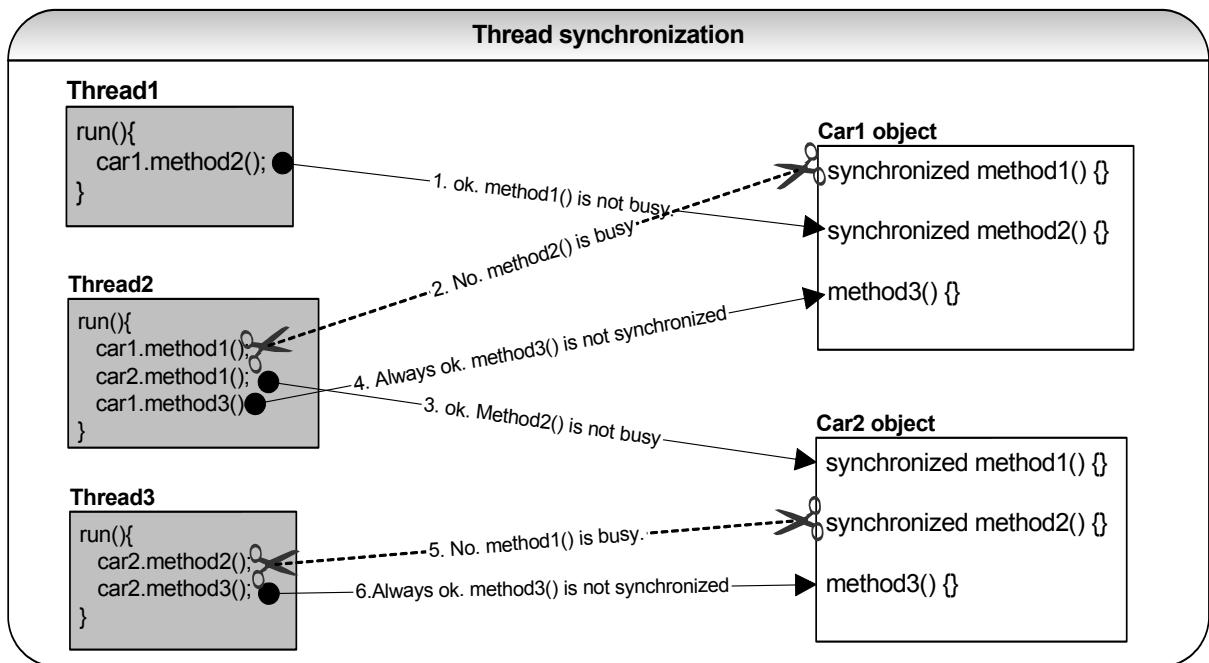
 public synchronized void consume() {
 while(count == 0) {
 try {
 wait();
 } catch(InterruptedException ie) {
 // keep trying
 }
 }
 count--; // consumed
 }

 private synchronized void produce() {
 count++;
 notify(); // notify the consumer that count has been incremented.
 }
}

```

**Q 43:** If 2 different threads hit 2 different synchronized methods in an object at the same time will they both continue? **LF**

**A 43:** No. Only one method can acquire the lock.



**Q 44:** Explain threads blocking on I/O? LF

**A 44:** Occasionally threads have to block on conditions other than object locks. I/O is the best example of this. Threads block on I/O (i.e. enters the waiting state) so that other threads may execute while the I/O operation is performed. When threads are blocked (say due to time consuming reads or writes) on an I/O call inside an object's synchronized method and also if the other methods of the object are also synchronized then the object is essentially frozen while the thread is blocked.

**Be sure to not synchronize code that makes blocking calls,** or make sure that a non-synchronized method exists on an object with synchronized blocking code. Although this technique requires some care to ensure that the resulting code is still thread safe, it allows objects to be responsive to other threads when a thread holding its locks is blocked.

**Note:** The `java.nio.*` package was introduced in JDK1.4. The coolest addition is nonblocking I/O (aka NIO that stands for New I/O). Refer **Q20** in Java section for NIO.

**Note:** Q45 & Q46 are very popular questions on design patterns.

**Q 45:** What is a **singleton** pattern? How do you code it in Java? DP MI CO

**A 45:** A singleton is a class that can be instantiated **only one time in a JVM per class loader**. Repeated calls always return the same instance. Ensures that a class has only one instance, and provide a **global point of access**. It can be an issue if singleton class gets loaded by multiple class loaders.

```
public class OnlyOne {
 private static OnlyOne one = new OnlyOne();

 private OnlyOne(){...} //private constructor. This class cannot be instantiated from outside.

 public static OnlyOne getInstance() {
 return one;
 }
}
```

**To use it:**

//No matter how many times you call, you get the same instance of the object.

```
OnlyOne myOne = OnlyOne.getInstance();
```

**Note:** The constructor must be explicitly declared and should have the private access modifier, so that it cannot be instantiated from outside the class. The only way to instantiate an instance of class *OnlyOne* is through the *getInstance()* method with a public access modifier.

**When to use:** Use it when only a single instance of an object is required in memory for a single point of access. For example the following situations require a **single point of access**, which gets invoked from various parts of the code.

- Accessing application specific properties through a singleton object, which reads them for the first time from a properties file and subsequent accesses are returned from in-memory objects. Also there could be another piece of code, which periodically synchronizes the in-memory properties when the values get modified in the underlying properties file. This piece of code accesses the in-memory objects through the singleton object (i.e. global point of access).
- Accessing in-memory object cache or object pool, or non-memory based resource pools like sockets, connections etc through a singleton object (i.e. global point of access).

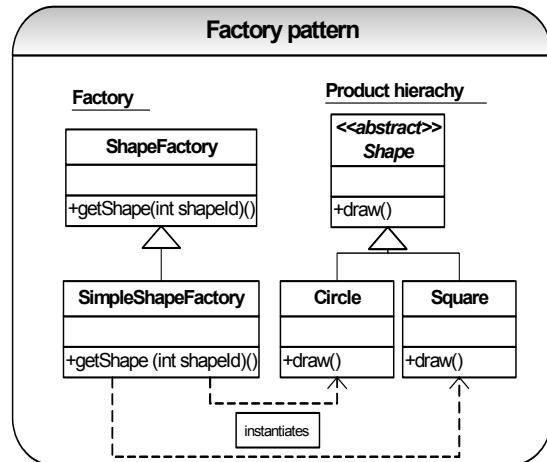
**What is the difference between a singleton class and a static class?** Static class is one approach to make a class singleton by declaring the class as "final" so that it cannot be extended and declaring all the methods as static so that you can't create any instance of the class and can call the static methods directly.

**Q 46:** What is a factory pattern? **DP CO**

**A 46:** A **Factory method pattern** (aka **Factory pattern**) is a creational pattern. The creational patterns abstract the object instantiation process by hiding how the objects are created and make the system independent of the object creation process. An **Abstract factory** pattern is one level of abstraction higher than a factory method pattern, which means it returns the factory classes.

#### Factory method pattern (aka Factory pattern)

Factory for what? Factory pattern returns one of the several product subclasses. You should use a factory pattern if you have a super class and a number of subclasses, and based on some data provided, you have to return the object of one of the subclasses. Let's look at a sample code:



```

public interface Const {
 public static final int SHAPE_CIRCLE = 1;
 public static final int SHAPE_SQUARE = 2;
 public static final int SHAPE_HEXAGON = 3;
}

public class ShapeFactory {
 public abstract Shape getShape(int shapeId);
}

public class SimpleShapeFactory extends ShapeFactory throws BadShapeException {
}

```

#### Abstract factory pattern

An **Abstract factory** pattern is one level of abstraction higher than a factory method pattern, which means the **abstract factory returns the appropriate factory classes**, which will later on return one of the product subclasses. Let's look at a sample code:

```

public class ComplexShapeFactory extends ShapeFactory {
 throws BadShapeException {
 public Shape getShape(int shapeTypeId) {
 Shape shape = null;
 if(shapeTypeId == Const.SHAPE_HEXAGON) {
 shape = new Hexagon(); //complex shape
 } else throw new BadShapeException
 ("shapeTypeId=" + shapeTypeId);
 return shape;
 }
}

```

Now let's look at the abstract factory, which returns one of the types of ShapeFactory:

```

public class ShapeFactoryType
 throws BadShapeFactoryException {

 public static final int TYPE_SIMPLE = 1;
 public static final int TYPE_COMPLEX = 2;

 public ShapeFactory getShapeFactory(int type) {
 ShapeFactory sf = null;
 if(type == TYPE_SIMPLE) {
 sf = new SimpleShapeFactory();
 } else if (type == TYPE_COMPLEX) {
 sf = new ComplexShapeFactory();
 } else throw new BadShapeFactoryException("No factory!!");
 }
}

```

<pre> public Shape getShape(int shapeTypeId){     Shape shape = null;     if(shapeTypeId == Const.SHAPE_CIRCLE) {         //in future can reuse or cache objects.         shape = new Circle();     }     else if(shapeTypeId == Const.SHAPE_SQUARE) {         //in future can reuse or cache objects         shape = new Square();     }     else throw new BadShapeException         ("ShapeTypeId=" + shapeTypeId);      return shape; } </pre>	<pre> return sf; }  Now let's look at the calling code, which uses the factory:  ShapeFactoryType abFac = new ShapeFactoryType(); ShapeFactory factory = null; Shape s = null;  //returns a ShapeFactory but whether it is a //SimpleShapeFactory or a ComplexShapeFactory is not //known to the caller. factory = abFac.getShapeFactory(1); //returns SimpleShapeFactory //returns a Shape but whether it is a Circle or a Pentagon is //not known to the caller. s = factory.getShape(2); //returns square. s.draw(); //draws a square  //returns a ShapeFactory but whether it is a //SimpleShapeFactory or a ComplexShapeFactory is not //known to the caller. factory = abFac.getShapeFactory(2); //returns a Shape but whether it is a Circle or a Pentagon is //not known to the caller. s = factory.getShape(3); //returns a pentagon. s.draw(); //draws a pentagon </pre>
<p>Now let's look at the calling code, which uses the factory:</p> <pre> ShapeFactory factory = new SimpleShapeFactory();  //returns a Shape but whether it is a Circle or a //Square is not known to the caller. Shape s = factory.getShape(1); s.draw(); // circle is drawn  //returns a Shape but whether it is a Circle or a //Square is not known to the caller. s = factory.getShape(2); s.draw(); //Square is drawn </pre>	

**Why use factory pattern or abstract factory pattern?** Factory pattern returns an instance of several (product hierarchy) subclasses (like **Circle**, **Square** etc), but the calling code is unaware of the actual implementation class. The calling code invokes the method on the interface (for example **Shape**) and using polymorphism the correct **draw()** method gets invoked [Refer Q8 in Java section for polymorphism]. So, as you can see, the factory pattern reduces the coupling or the dependencies between the calling code and called objects like **Circle**, **Square** etc. This is a very powerful and common feature in many frameworks. You do not have to create a new **Circle** or a new **Square** on each invocation as shown in the sample code, which is for the purpose of illustration and simplicity. In future, to conserve memory you can decide to cache objects or reuse objects in your factory with no changes required to your calling code. You can also load objects in your factory based on attribute(s) read from an external properties file or some other condition. Another benefit going for the factory is that unlike calling constructors directly, factory patterns have more meaningful names like **getShape(...)**, **getInstance(...)** etc, which may make calling code more clear.

**Can we use the singleton pattern within our factory pattern code?** Yes. Another important aspect to consider when writing your factory class is that, it does not make sense to create a new factory object for each invocation as it is shown in the sample code, which is just fine for the illustration purpose.

```
ShapeFactory factory = new SimpleShapeFactory();
```

To overcome this, you can incorporate the singleton design pattern into your factory pattern code. The singleton design pattern will create only a single instance of your **SimpleShapeFactory** class. Since an abstract factory pattern is unlike factory pattern, where you need to have an instance for each of the two factories (i.e. **SimpleShapeFactory** and **ComplexShapeFactory**) returned, you can still incorporate the singleton pattern as an access point and have an instance of a **HashMap**, store your instances of both factories. Now your calling method uses a static method to get the same instance of your factory, hence conserving memory and promoting object reuse:

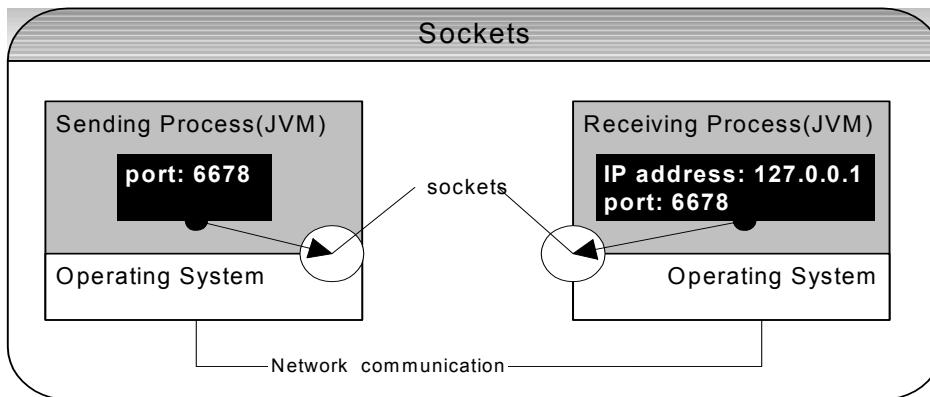
```
ShapeFactory factory = ShapeFactory.getInstance();
factory.getShape();
```

**Note:** Since questions on singleton pattern and factory pattern are commonly asked in the interviews, they are included as part of this section. To learn more about design patterns refer Q11 in How would you go about section...?

**A 47:** A socket is a communication channel, which facilitates **inter-process communication** (For example communicating between two JVMs, which may or may not be running on two different physical machines). A socket is an endpoint for communication. There are two kinds of sockets, depending on whether one wishes to use a connectionless or a connection-oriented protocol. The connectionless communication protocol of the Internet is called UDP. The connection-oriented communication protocol of the Internet is called TCP. UDP sockets are also called datagram sockets. Each socket is uniquely identified on the entire Internet with two numbers. The first number is a 128-bit integer called the Internet Address (or **IP address**). The second number is a 16-bit integer called the **port** of the socket. The IP address is the location of the machine, which you are trying to connect to and the port number is the port on which the server you are trying to connect is running. The port numbers 0 to 1023 are reserved for standard services such as e-mail, FTP, HTTP etc.

The lifetime of the socket is made of 3 phases: **Open Socket → Read and Write to Socket → Close Socket**

To make a socket connection you need to know two things: An IP address and port on which to listen/connect. In Java you can use the **Socket** (client side) and **ServerSocket** (Server side) classes.



**Q 48:** How will you call a Web server from a stand alone Java application? **LF**

**A 48:** Using the **java.net.URLConnection** and its subclasses like **HttpURLConnection** and **JarURLConnection**.

URLConnection	HttpClient (browser)
Supports HEAD, GET, POST, PUT, DELETE, TRACE and OPTIONS	Supports HEAD, GET, POST, PUT, DELETE, TRACE and OPTIONS.
Does not support cookies.	Does support cookies.
Can handle protocols other than http like ftp, gopher, mailto and file.	Handles only http.

## Java – Swing

**Q 49:** What is the difference between AWT and Swing? **LF DC**

**A 49:** Swing provides a richer set of components than AWT. They are 100% Java-based. There are a few other advantages to Swing over AWT:

- Swing provides both additional components like JTable, JTree etc and added functionality to AWT-replacement components.
- Swing components can change their appearance based on the current “look and feel” library that’s being used.
- Swing components follow the **Model-View-Controller** (MVC) paradigm, and thus can provide a much more flexible UI.
- Swing provides “extras” for components, such as: icons on many components, decorative borders for components, tool tips for components etc.
- Swing components are lightweight (less resource intensive than AWT).

- Swing provides built-in **double buffering** (which means an off-screen buffer [image] is used during drawing and then the resulting bits are copied onto the screen. The resulting image is smoother, less flicker and quicker than drawing directly on the screen).
- Swing provides paint debugging support for when you build your own component i.e.-slow motion rendering.

**Swing also has a few disadvantages:**

- If you're not very careful when programming, it can be slower than AWT (all components are drawn).
- Swing components that look like native components might not behave exactly like native components.

**Q 50:** Explain the Swing Action architecture? **LF DP**

**A 50:** The Swing Action architecture is used to implement shared behaviour between two or more user interface components. For example, the menu items and the tool bar buttons will be performing the same action no matter which one is clicked. Another distinct advantage of using actions is that when an action is disabled then all the components, which use the Action, become disabled.

**Design pattern:** The `javax.swing.Action` interface extends the `ActionListener` interface and is an abstraction of a command that does not have an explicit UI component bound to it. The Action architecture is an implementation of a **command design pattern**. This is a powerful design pattern because it allows the separation of controller logic of an application from its visual representation. This allows the application to be easily configured to use different UI elements without having to re-write the control or call-back logic.

Defining action classes:

```
class FileAction extends AbstractAction {
 //Constructor
 FileAction(String name) {
 super(name);
 }

 public void actionPerformed(ActionEvent ae){
 //add action logic here
 }
}
```

To add an action to a menu bar:

```
JMenu fileMenu = new JMenu("File");
FileAction newAction = new FileAction("New");
JMenuItem item = fileMenu.add(newAction);
item.setAccelaror(KeyStroke.getKeyStroke('N', Event.CTRL_MASK));
```

To add action to a toolbar

```
private JToolBar toolbar = new JToolBar();
toolbar.add(newAction);
```

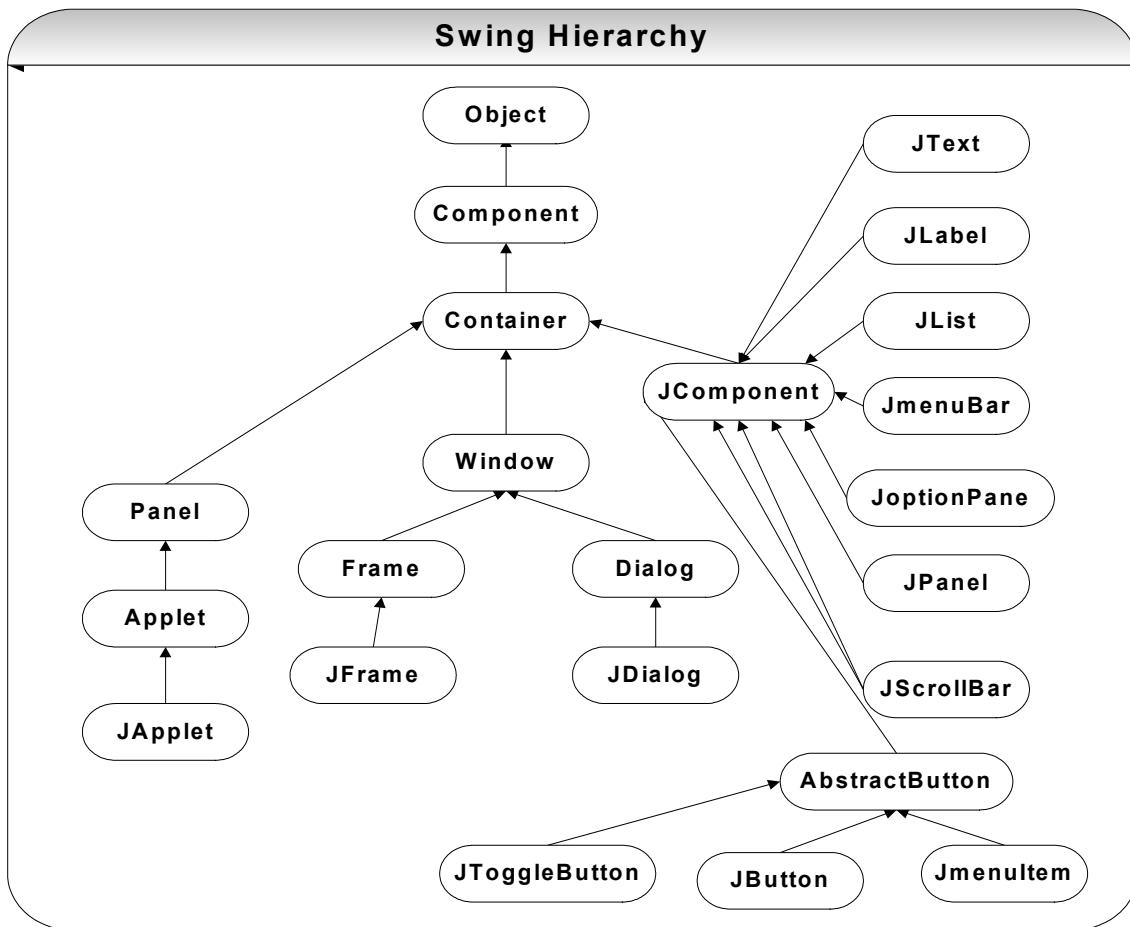
So, an *action* object is a listener as well as an action.

**Q 51:** If you add a component to the CENTER of a border layout, which directions will the component stretch? **LF**

**A 51:** The component will stretch both horizontally and vertically. It will occupy the whole space in the middle.

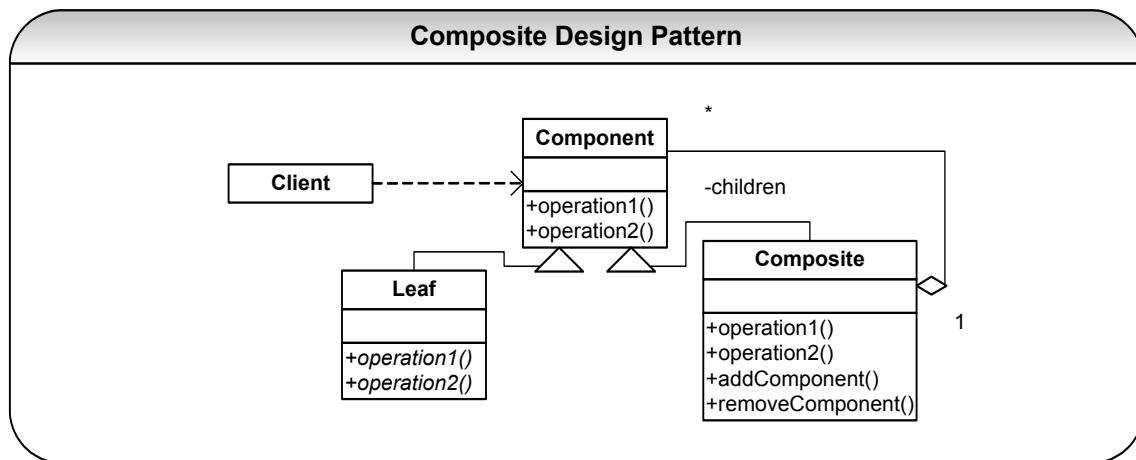
**Q 52:** What is the base class for all Swing components? **LF**

**A 52:** All the Swing components start with 'J'. The hierarchy diagram is shown below. **JComponent** is the base class.



(Diagram source: <http://www.particle.kth.se/~fmi/kurs/PhysicsSimulation/Lectures/07A/swingDesign.html>)

**Design pattern:** As you can see from the above diagram, containers collect components. Sometimes you want to add a container to another container. So, a container should be a component. For example `container.getPreferredSize()` invokes `getPreferredSize()` of all contained components. **Composite design pattern** is used in GUI components to achieve this. A composite object is an object, which contains other objects. Composite design pattern manipulates composite objects just like you manipulate individual components. Refer Q11 in How would you go about...? section.



**Q 53:** Explain the Swing event dispatcher mechanism? **[LF CI PI]**

**A 53:** Swing components can be accessed by the Swing **event dispatching thread**. A few operations are guaranteed to be thread-safe but most are not. Generally the Swing components should be accessed through this **event-**

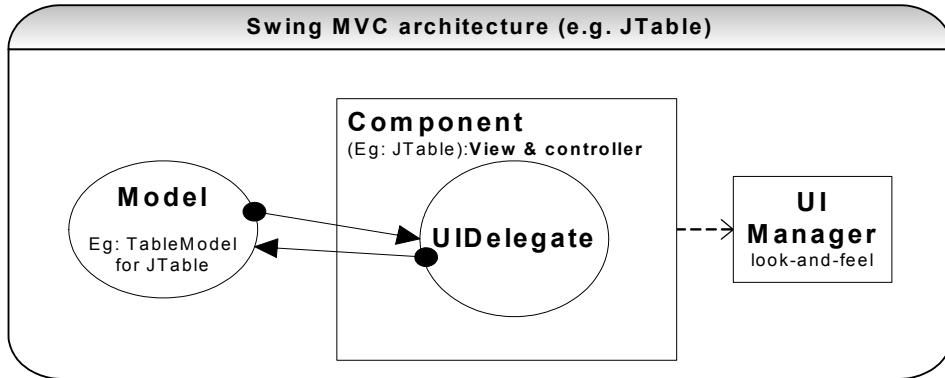
**dispatching thread.** The *event-dispatching* thread is a thread that executes drawing of components and event-handling code. For example the `paint()` and `actionPerformed()` methods are automatically executed in the *event-dispatching thread*. Another way to execute code in the *event-dispatching thread* from outside event-handling or drawing code, is using `SwingUtilities.invokeLater()` or `invokeAndWait()` method. **Swing lengthy initialization tasks (e.g. I/O bound and computationally expensive tasks), should not occur in the event-dispatching thread because this will hold up the dispatcher thread.** If you need to create a new thread for example, to handle a job that's computationally expensive or I/O bound then you can use the thread utility classes such as `SwingWorker` or `Timer` without locking up the *event-dispatching thread*.

- **SwingWorker** – creates a background thread to execute time consuming operations.
- **Timer** – creates a thread that executes at certain intervals.

However after the lengthy initialization the GUI update should occur in the event dispatching thread, for thread safety reasons. We can use `invokeLater()` to execute the GUI update in the *event-dispatching thread*. The other scenario where `invokeLater()` will be useful is that the GUI must be updated as a result of non-AWT event.

**Q 54:** What do you understand by MVC as used in a JTable? **LF DP**

**A 54:** MVC stands for Model View Controller architecture. Swing "J" components (e.g. `JTable`, `JList`, `JTree` etc) use a modified version of MVC. MVC separates a model (or data source) from a presentation and the logic that manages it.



- **Component** (e.g. `JTable`, `JTree`, and `JList`): coordinates actions of model and the UI delegate. Each generic component class handles its own individual **view-and-controller** responsibilities.
- **Model** (e.g. `TableModel`): charged with storing the data.
- **UIDelegate**: responsible for getting the data from model and rendering it to screen. It delegates any look-and-feel aspect of the component to the **UI Manager**.

**Q 55:** Explain layout managers? **LF**

**A 55:** Layout managers are used for arranging GUI components in windows. The standard layout managers are:

- **FlowLayout:** Default layout for **Applet** and **Panel**. Lays out components from left to right, starting new rows if necessary.
- **BorderLayout:** Default layout for **Frame** and **Dialog**. Lays out components in north, south, east, west and center. All extra space is placed on the center.
- **CardLayout:** stack of same size components arranged inside each other. Only one is visible at any time. Used in TABs.
- **GridLayout:** Makes a bunch of components equal in size and displays them in the requested number of rows and columns.
- **GridBagLayout:** Most complicated but the most flexible. It aligns components by placing them within a grid of cells, allowing some components to span more than one cell. The rows in the grid aren't necessarily all the same height, similarly, grid columns can have different widths as well.

- **BoxLayout:** is a full-featured version of FlowLayout. It stacks the components on top of each other or places them in a row.

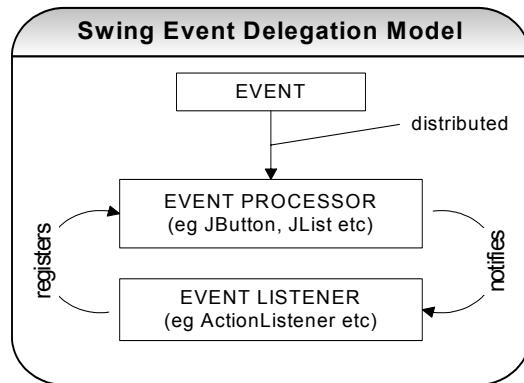
Complex layouts can be simplified by using nested containers for example having *panels* within *panels* and each *panel* can use its own *LayoutManager*. It is also possible to write your own layout manager or use manual positioning of the GUI components. **Note:** Further reading on each *LayoutManager*s is recommended for Swing developers.

**Design pattern:** The AWT containers like panels, dialog boxes, windows etc do not perform the actual laying out of the components. They delegate the layout functionality to layout managers. The layout managers make use of the **strategy design pattern**, which encapsulates family of algorithms for laying out components in the containers. If a particular layout algorithm is required other than the default algorithm, an appropriate layout manager can be instantiated and plugged into the container (e.g. panels by default uses the FlowLayout but it can be changed by executing → panel.setLayout(new GridLayout(4,5))). This enables the layout algorithms to vary independently from the containers that uses them, this is one of the key benefits of the strategy pattern.

**Q 56:** Explain the Swing delegation event model? **LF**

**A 56:** In this model, the objects that receive user events notify the registered listeners of the user activity. In most cases the event receiver is a component.

- **Event Types:** ActionEvent, KeyEvent, MouseEvent, WindowEvent etc.
- **Event Processors:** JButton, JList etc.
- **EventListeners:** ActionListener, ComponentListener, KeyListener etc.



#### Java – Applet

**Q 57:** How will you initialize an applet? **LF**

**A 57:** By writing your initialization code in the applet's **init()** method or applet's **constructor**.

**Q 58:** What is the order of method invocation in an applet? **LF**

**A 58:** The Applet's life cycle methods are as follows:

- **public void init():** Initialization method called only once by the browser.
- **public void start():** Method called after init() and contains code to start processing. If the user leaves the page and returns without killing the current browser session, the start () method is called without being preceded by init ().
- **public void stop():** Stops all processing started by start (). Done if user moves off page.
- **public void destroy():** Called if current browser session is being terminated. Releases all resources used by the applet.

**Q 59:** How would you communicate between applets and servlets? **LF**

**A 59:** We can use the **java.net.URLConnection** and **java.net.URL** classes to open a standard HTTP connection and “tunnel” to a Web server. The server then passes this information to the servlet. Basically, the applet pretends to be a Web browser, and the servlet doesn’t know the difference. As far as the servlet is concerned, the applet is just another HTTP client. Applets can communicate with servlets using GET or POST methods.

The parameters can be passed between the applet and the servlet as **name value pairs**.

```
http://www.foo.com/servlet/TestServlet?LastName=Jones&FirstName=Joe).
```

**Objects** can also be passed between applet and servlet using object serialization. Objects are serialized to and from the **inputstream** and **outputstream** of the connection respectively.

**Q 60:** How will you communicate between two Applets? **LF**

**A 60:** All the applets on a given page share the same AppletContext. We obtain this applet context as follows:

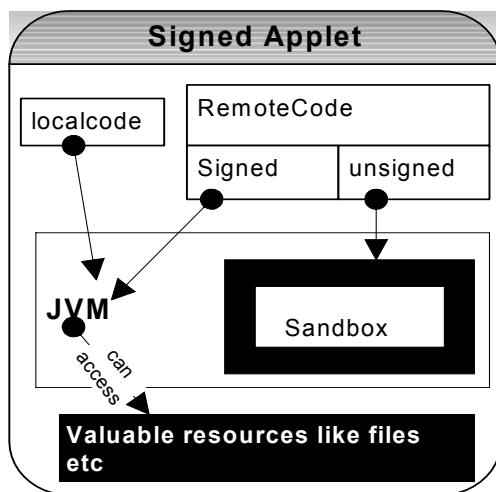
```
AppletContext ac = getAppletContext();
```

AppletContext provides applets with methods such as **getApplet(name)**, **getApplets()**, **getAudioClip**, **getImage**, **showDocument** and **showStatus()**.

**Q 61:** What is a signed Applet? **LF SE**

**A 61:** A signed Applet is a **trusted** Applet. By default, and for security reasons, Java applets are contained within a “**sandbox**”. Refer to the diagram below:

This means that the applets can't do anything, which might be construed as threatening to the user's machine (e.g. reading, writing or deleting local files, putting up message windows, or querying various system parameters). Early browsers had no provisions for Java applets to reach outside of the sandbox. Recent browsers, however (Internet Explorer 4 on Windows etc), have provisions to give “**trusted**” applets the ability to work outside the sandbox. For this power to be granted to one of your applets, the applet's code must be digitally signed with your unforgeable digital ID, and then the user must state that he trusts applets signed with your ID. The untrusted applet can request to have privileges outside the sand box but will have to request the user for privileges every time it executes. But with the trusted applet the user can choose to remember their answer to the request, which means they won't be asked again.



**Q 62:** What is the difference between an applet and an application? **LF**

**A 62:**

Applet	Application
Applets don't have a main method. They operate on life cycle methods <code>init()</code> , <code>start()</code> , <code>stop()</code> , <code>destroy()</code> etc.	Has a static <code>main()</code> method.
Applets can be embedded in HTML pages and	Has no support for embedding or downloading. Has

downloaded over the Internet. Has a sand box security model.	no inherent security restriction.
Can only be executed within a Java compatible container like browser, appletviewer etc.	Applications are executed at command line by java.exe.

### Java – Performance and Memory leaks

**Q 63:** How would you improve performance of a Java application? **P|B|P**

**A 63:**

- **Pool valuable system resources** like threads, database connections, socket connections etc. Emphasise on reuse of threads from a pool of threads. Creating new threads and discarding them after use can adversely affect performance. Also consider using multi-threading in your single-threaded applications where possible to enhance performance. Optimise the pool sizes based on system and application specifications and requirements.
- **Optimize your I/O operations:** use buffering (Refer **Q21** in Java section) when writing to and reading from files and/or streams. Avoid writers/readers if you are dealing with only ASCII characters. You can use streams instead, which are faster. Avoid premature flushing of buffers. Also make use of the performance and scalability enhancing features such as non-blocking and asynchronous I/O, mapping of file to memory etc offered by the NIO (New I/O).
- **Minimize network overheads** by retrieving several related items simultaneously in one remote invocation if possible. Remote method invocations involve a network round-trip, marshalling and unmarshalling of parameters, which can cause huge performance problems if the remote interface is poorly designed. (Refer **Q125** in Enterprise section).
- **Establish whether you have a potential memory problem and manage your objects efficiently:** remove references to the short-lived objects from long-lived objects like Java collections etc (Refer **Q64** in Java section) to minimise any potential memory leaks. Also reuse objects where possible. It is cheaper to recycle objects than creating new objects each time. Avoid creating extra objects unnecessarily. For example use mutable *StringBuffer/StringBuilder* classes instead of immutable *String* objects in computation expensive loops as discussed in **Q17** in Java section. Automatic garbage collection is one of the most highly touted conveniences of Java. However, it comes at a price. Creating and destroying objects occupies a significant chunk of the JVM's time. Wherever possible, you should look for ways to minimise the number of objects created in your code:
  - If repeating code within a loop, avoid creating new objects for each iteration. Create objects before entering the loop (i.e. outside the loop) and reuse them if possible.
  - For complex objects that are used frequently, consider creating a pool of recyclable objects rather than always instantiating new objects. This adds additional burden on the programmer to manage the pool, but in select cases can represent an order of magnitude performance gain.
  - Use lazy initialization when you want to distribute the load of creating large amounts of objects. Use lazy initialization only when there is merit in the design.
- **Where applicable apply the following performance tips in your code:**
  - Use ArrayLists, HashMap etc as opposed to Vector, Hashtable etc where possible. This is because the methods in ArrayList, HashMap etc are not synchronized (Refer **Q13** in Java Section). Even better is to use just arrays where possible.
  - Set the initial capacity of a collection (e.g. *ArrayList*, *HashMap* etc) and *StringBuffer/StringBuilder* appropriately. This is because these classes must grow periodically to accommodate new elements. So, if you have a very large *ArrayList* or a *StringBuffer*, and you know the size in advance then you can speed things up by setting the initial size appropriately. (Refer **Q15**, **Q17** in Java Section).

- Minimise the use of **casting** or runtime type checking like ***instanceof*** in frequently executed methods or in loops. The “casting” and “instanceof” checks for a class marked as final will be faster. Using “instanceof” construct is not only ugly but also unmaintainable. Look at using **visitor pattern** (Refer **Q11** in How would you go about...? section) to avoid “instanceof” construct.
- Do not compute constants inside a large loop. Compute them outside the loop. For applets compute it in the init() method.
- Exception creation can be expensive because it has to create the full stack trace. The stack trace is obviously useful if you are planning to log or display the exception to the user. But if you are using your exception to just control the flow, which is not recommended, then throw an exception, which is pre-created. An efficient way to do this is to declare a public static final **Exception** in your exception class itself.
- Avoid using System.out.println and use logging frameworks like Log4J etc, which uses I/O buffers (Refer **Q21** in Java section).
- Minimise calls to Date, Calendar, etc related classes.
- Minimise JNI calls in your code.

**Note:** Set performance requirements in the specifications, include a performance focus in the analysis and design and also create a performance test environment.

**Q 64:** How would you detect and minimise memory leaks in Java? **MI BP**

**A 64:** In Java memory leaks are caused by poor program design where object references are long lived and the garbage collector is unable to reclaim those objects.

#### Detecting memory leaks:

- Use tools like JProbe, Optimizelt etc to detect memory leaks.
- Use operating system process monitors like task manager on NT systems, ps, vmstat, iostat, netstat etc on UNIX systems.
- Write your own utility class with the help of totalMemory() and freeMemory() methods in the Java *Runtime* class. Place these calls in your code strategically for pre and post memory recording where you suspect to be causing memory leaks. An even better approach than a utility class is using **dynamic proxies** (Refer **Q11** in How would you go about section...) or **Aspect Oriented Programming (AOP)** for pre and post memory recording where you have the control of activating memory measurement only when needed. (Refer **Q3 – Q5** in Emerging Technologies/Frameworks section).

#### Minimising memory leaks:

In Java, typically memory leak occurs when **an object of a longer lifecycle has a reference to objects of a short life cycle**. This prevents the objects with short life cycle being garbage collected. The developer must remember to remove the references to the short-lived objects from the long-lived objects. Objects with the same life cycle do not cause any issues because the garbage collector is smart enough to deal with the circular references (Refer **Q33** in Java section).

- Design applications with an object's life cycle in mind, instead of relying on the clever features of the JVM. **Letting go** of the object's reference in one's own class as soon as possible can mitigate memory problems.  
**Example:** myRef = null;
- Unreachable collection objects can magnify a memory leak problem. In Java it is easy to let go of an entire collection by setting the root of the collection to null. The garbage collector will reclaim all the objects (unless some objects are needed elsewhere).
- Use weak references (Refer **Q32** in Java section) if you are the only one using it. The **WeakHashMap** is a combination of *HashMap* and *WeakReference*. This class can be used for programming problems where you need to have a *HashMap* of information, but you would like that information to be garbage collected if you are the only one referencing it.

- Free native system resources like AWT frame, files, JNI etc when finished with them. **Example:** Frame, Dialog, and Graphics classes require that the method dispose() be called on them when they are no longer used, to free up the system resources they reserve.

**Q 65:** Why does the JVM crash with a core dump or a Dr.Watson error? **M1**

**A 65:** Any problem in pure Java code throws a Java exception or error. Java exceptions or errors will not cause a core dump (on UNIX systems) or a Dr.Watson error (on WIN32systems). Any serious Java problem will result in an **OutOfMemoryError** thrown by the JVM with the stack trace and consequently JVM will exit. These Java stack traces are very useful for identifying the cause for an abnormal exit of the JVM. So is there a way to know that **OutOfMemoryError** is about to occur? The Java JDK 1.5 has a package called java.lang.management which has useful JMX beans that we can use to manage the JVM. One of these beans is the MemoryMXBean.

An **OutOfMemoryError** can be thrown due to one of the following 4 reasons:

- JVM may have a memory leak due to a bug in its internal heap management implementation. But this is highly unlikely because JVMs are well tested for this.
- The application may not have enough heap memory allocated for its running. You can allocate more JVM heap size (with -Xmx parameter to the JVM) or decrease the amount of memory your application takes to overcome this. To increase the heap space:

```
Java -Xms1024M -Xmx1024M
```

Care should be taken not to make the -Xmx value too large because it can slow down your application. The secret is to make the maximum heap size value the right size.

- Another not so prevalent cause is the running out of a memory area called the “perm” which sits next to the heap. All the binary code of currently running classes is archived in the “perm” area. The ‘perm’ area is important if your application or any of the third party jar files you use dynamically generate classes. **For example:** “perm” space is consumed when XSLT templates are dynamically compiled into classes, J2EE application servers, JasperReports, JAXB etc use Java reflection to dynamically generate classes and/or large amount of classes in your application. To increase perm space:

```
Java -XX:PermSize=256M -XX:MaxPermSize=256M
```

- The fourth and the most common reason is that you may have a memory leak in your application as discussed in **Q64** in Java section.

[Good read/reference: “**Know Your Worst Friend, the Garbage Collector**” <http://java.sys-con.com/read/84695.htm> by Romain Guy]

### So why does the JVM crash with a core dump or Dr.Watson error?

Both the core dump on UNIX operating system and Dr.Watson error on WIN32 systems mean the same thing. The JVM is a process like any other and when a process crashes a core dump is created. A core dump is a memory map of a running process. This can happen due to one of the following reasons:

- Using JNI (Java Native Interface) code, which has a fatal bug in its native code. **Example:** using Oracle OCI drivers, which are written partially in native code or jdbc-odbc bridge drivers, which are written in non Java code. Using 100% pure Java drivers (communicates directly with the database instead of through client software utilizing the JNI) instead of native drivers can solve this problem. We can use Oracle thin driver, which is a 100% pure Java driver.
- The operating system on which your JVM is running might require a patch or a service pack.
- The JVM implementation you are using may have a bug in translating system resources like threads, file handles, sockets etc from the platform neutral Java byte code into platform specific operations. If this JVM’s translated native code performs an illegal operation then the **operating system will instantly kill the process and mostly will generate a core dump file**, which is a hexadecimal file indicating program’s state in memory at the time of error. The core dump files are generated by the operating system in response to certain signals. Operating system signals are responsible for notifying certain events to its threads and processes. The JVM can also intercept certain signals like **SIGQUIT** which is kill -3 < process id > from the operating system and it responds to this signal by printing out a Java stack trace and then continue to run.

The JVM continues to run because the JVM has a special built-in debug routine, which will trap the **signal -3**. On the other hand signals like **SIGSTOP** (kill -23 <process id>) and **SIGKILL** (kill -9 <process id>) will cause the JVM process to stop or die. The following JVM argument will indicate JVM not to pause on **SIGQUIT** signal from the operating system.

#### Java –Xsqnopause

#### Java – Personal

**Q 66:** Did you have to use any design patterns in your Java project? **DP**

**A 66:** Yes. Refer **Q10 [Strategy]**, **Q14 [Iterator]**, **Q20 [Decorator]**, **Q31 [Visitor]**, **Q45 [Singleton]**, **Q46 [Factory]**, **Q50 [Command]**, and **Q54 [MVC]** in Java section and **Q11** in How would you go about... section. Note: Learning of other patterns recommended (Gang of Four Design Patterns).

**Resource:** <http://www.patterndepot.com/put/8/JavaPatterns.htm>.

Why use design patterns, you may ask (Refer **Q5** in Enterprise section). Design patterns are worthy of mention in your CV and interview. Design patterns have a number of advantages:

- Capture design experience from the past.
- Promote reuse without having to reinvent the wheel.
- Define the system structure better.
- Provide a common design vocabulary.

**Some advice if you are just starting on your design pattern journey:**

- If you are not familiar with UML, now is the time. UML is commonly used to describe patterns in pattern catalogues, including class diagrams, sequence diagrams etc. (Refer **Q106 - Q109** in Enterprise section).

**Class diagrams** describe the types of objects in the system and the various static relationships among them. Class diagrams also show the attributes and the methods. When using patterns, it is important to define a naming convention. It will be much easier to manage a project as it grows to identify exactly what role an object plays with the help of a naming convention e.g. AccountFacilityBusinessDelegate, AccountFacilityFactory, AccountFacilityValueObject, AccountDecorator, AccountVisitor, AccountTransferObject (or AccountFacilityVO or AccountTO).

- Make a list of requirements that you will be addressing and then try to identify relevant patterns that are applicable.

**Sequence diagrams** are interaction diagrams which detail what messages are sent and when. The sequence diagrams are organized according to time. The time progresses as you move from top to bottom of the diagram. The objects involved in the diagram are shown from left to right according to when they take part.

**Q 67:** Tell me about yourself or about some of the recent projects you have worked with? What do you consider your most significant achievement? Why do you think you are qualified for this position? Why should we hire you and what kind of contributions will you make?

**A 67:** [Hint:] Pick your recent projects and brief on it. Also is imperative that during your briefing, you demonstrate how you applied your skills and knowledge in some of the following areas:

- Design concepts and design patterns: **How you understand and applied them.**
- Performance and memory issues: **How you identified and fixed them.**
- Exception handling and best practices: **How you understand and applied them.**
- Multi-threading and concurrent access: **How you identified and fixed them.**

Some of the questions in this section can help you prepare your answers by relating them to your current or past work experience. For example:

- **Design Concepts:** Refer **Q5, Q6, Q7, Q8, Q9** etc
- **Design Patterns:** Refer **Q10, Q14, Q20, Q31, Q45, Q46, Q50** etc [Refer **Q11** in How would you go about...? section]
- **Performance issues:** Refer **Q21, Q63** etc
- **Memory issues:** Refer **Q32, Q64, Q65** etc
- **Exception Handling:** Refer **Q34, Q35** etc
- **Multi-threading (Concurrency issues):** Refer **Q29, Q40** etc

Demonstrating your knowledge in the above mentioned areas will improve your chances of being successful in your Java/J2EE interviews. 90% of the interview questions are asked based on your own resume. So in my view it is also very beneficial to mention how you demonstrated your knowledge/skills by stepping through a recent project on your resume.

The two other areas, which I have not mentioned in this section, which are also very vital, are transactions and security. These two areas will be covered in the next section, which is the Enterprise section (J2EE, JDBC, EJB, JMS, SQL, XML etc).

Even if you have not applied these skills knowingly or you have not applied them at all, just demonstrating that you have the knowledge and an appreciation will help you improve your chances in the interviews. Also mention any long hours worked to meet the deadline, working under pressure, fixing important issues like performance issues, running out of memory issues etc.

---

**Q 68:** Why are you leaving your current position?

**A 68:** [Hint]

- Do not criticize your previous employer or coworkers or sound too opportunistic.
  - It is fine to mention a major problem like a buy out, budget constraints, merger or liquidation.
  - You may also say that your chance to make a contribution is very low due to company wide changes or looking for a more challenging senior or designer role.
- 

**Q 69:** What do you like and/or dislike most about your current and/or last position?

**A 69:** [Hint]

The interviewer is trying to find the compatibility with the open position. So

**Do not say** anything like:

- You dislike overtime.
- You dislike management or coworkers etc.

It is **safe to say**:

- You like challenges.
  - Opportunity to grow into design, architecture, performance tuning etc.
  - You dislike frustrating situations like identifying a memory leak problem or a complex transactional or a concurrency issue. You want to get on top of it as soon as possible.
- 

**Q 70:** How do you handle pressure? Do you like or dislike these situations?

**A 70:** [Hint]

These questions could mean that the open position is pressure-packed and may be out of control. Know what you are getting into. If you do perform well under stress then give a descriptive example. High achievers tend to perform well in pressure situations.

---

**Q 71:** What are your strengths and weaknesses? Can you describe a situation where you took initiative? Can you describe a situation where you applied your problem solving skills?

**A 71:** [Hint]

**Strengths:**

- **Taking initiatives and being pro-active:** You can illustrate how you took initiative to fix a transactional issue, a performance problem or a memory leak problem.
- **Design skills:** You can illustrate how you designed a particular application using OO concepts.
- **Problem solving skills:** Explain how you will break a complex problem into more manageable sub-sections and then apply brain storming and analytical skills to solve the complex problem. Illustrate how you went about identifying a scalability issue or a memory leak problem.

- **Communication skills:** Illustrate that you can communicate effectively with all the team members, business analysts, users, testers, stakeholders etc.
- **Ability to work in a team environment as well as independently:** Illustrate that you are technically sound to work independently as well as have the interpersonal skills to fit into any team environment.
- **Hard working, honest, and conscientious etc** are the adjectives to describe you.

**Weaknesses:**

Select a trait and come up with a solution to overcome your weakness. Stay away from personal qualities and concentrate more on professional traits for example:

- I pride myself on being an attention to detail guy but sometimes miss small details. So I am working on applying the 80/20 principle to manage time and details. Spend 80% of my effort and time on 20% of the tasks, which are critical and important to the task at hand.
- Some times when there is a technical issue or a problem I tend to work continuously until I fix it without having a break. But what I have noticed and am trying to practise is that taking a break away from the problem and thinking outside the square will assist you in identifying the root cause of the problem sooner.

---

**Q 72:** What are your career goals? Where do you see yourself in 5-10 years?

**A 72:** [Hint] Be realistic. For example

- Next 2-3 years to become a senior developer or a team lead.
- Next 3-5 years to become a solution designer or an architect.

---

**Note:** For Q66 – Q72 tailor your answers to the job. Also be prepared for questions like:

- What was the last Java related book or article you read? [Hint]
  - **Mastering EJB** by Ed Roman.
  - **EJB design patterns** by Floyd Marinescu.
  - **Bitter Java** by Bruce Tate.
  - **Thinking in Java** by Bruce Eckel.
- Which Java related website(s) do you use to keep your knowledge up to date? [Hint]
  - <http://www.theserverside.com>
  - <http://www.javaworld.com>
  - <http://www-136.ibm.com/developerworks/Java>
  - <http://www.precisejava.com>
  - <http://www.allapplabs.com>
  - <http://java.sun.com>
  - <http://www.martinfowler.com>
  - <http://www.ambyssoft.com>
- What past accomplishments gave you satisfaction? What makes you want to work hard? [Hint]
  - Material rewards such as salary, perks, benefits etc naturally come into play but focus on your achievements or accomplishments than on rewards.
- Do you have any role models in software development? [Hint]
  - Scott W. Ambler, Martin Fowler, Ed Roman, Floyd Marinescu, Grady Booch etc.
- Why do you want to work for us? (Research the company prior to the interview).

## Java – Key Points

- Java is an object oriented (OO) language, which has built in support for multi-threading, socket communication, automatic memory management (i.e. garbage collection) and also has better portability than other languages across operating systems.
- Java class loaders are **hierarchical** and use a **delegation model**. The classes loaded by a child class loader have **visibility** into classes loaded by its parents up the hierarchy but the reverse is not true.
- Java does not support **multiple implementation inheritance** but supports **multiple interface inheritance**.
- **Polymorphism, inheritance and encapsulation** are the 3 pillar of an object-oriented language.
- Code reuse can be achieved through either **inheritance** ("is a" relationship) or **object composition** ("has a" relationship). Favour object composition over inheritance.
- When using **implementation inheritance**, make sure that the **subclasses depend only on the behaviour of the superclass**, not the actual implementation. An **abstract** base class usually provides an implementation inheritance.
- Favour **interface inheritance to implementation inheritance** because it promotes the design concept of **coding to interface** and **reduces coupling**. The interface inheritance can achieve code reuse through **object composition**.
- Design by contract specifies the obligations of a calling-method and called-method to each other using **pre-conditions, post-conditions** and **class invariants**.
- When using Java collection API, prefer using *ArrayList* or *HashMap* as opposed to *Vector* or *Hashtable* to **avoid any synchronization overhead**. The *ArrayList* or *HashMap* can be externally synchronized for concurrent access by multiple threads.
- Set the initial capacity of a collection appropriately and program in terms of interfaces as opposed to implementations.
- When providing a user defined key class for storing objects in *HashMap*, you should override **equals()**, and **hashCode()** methods from the *Object* class.
- *String* class is immutable and *StringBuffer* and *StringBuilder* classes are mutable. So it is more efficient to use a *StringBuffer* or a *StringBuilder* as opposed to a *String* in a computation intensive situations (ie. in for, while loops).
- **Serialization** is a process of writing an object to a file or a stream. **Transient** variables cannot be serialized.
- Java I/O performance can be improved by using buffering, minimising access to the underlying hard disk and operating systems. Use the NIO package for performance enhancing features like non-blocking I/O operation, buffers to hold data, and memory mapping of files.
- Each time an object is created in Java it goes into the area of memory known as **heap**. The primitive variables are allocated in the **stack** if they are local method variables and in the **heap** if they are class member variables.
- Threads **share the heap spaces** so it is **not thread-safe** and the threads have **their own stack space**, which is **thread-safe**.
- The **garbage collection cannot be forced**, but you can nicely ask the garbage collector to collect garbage.
- There two types of exceptions **checked** (ie compiler checked) and **unchecked** (Runtime Exceptions). It is not advisable to catch type *Exception*.
- A **process** is an execution of a program (e.g. JVM process) but a **thread** is a single execution sequence within the process.
- Threads can be created in Java by either extending the *Thread* class or implementing the *Runnable* interface.

- In Java each object has a lock and a thread can acquire a lock by using the **synchronized** key word. The synchronization key word can be applied in **method level** (coarse-grained lock) or **block level** (fine-grained lock which offers better performance) of code.
- Threads can communicate with each other using **wait()**, **notify()**, and **notifyAll()** methods. This communication solves the **consumer-producer** problem.
- Sockets are communication channels, which facilitate inter-process communication.
- Swing uses the **MVC paradigm** to provide loose coupling and action **architecture** to implement a shared behaviour between two or more user interface components.
- Swing components should be accessed through an **event-dispatching thread**. There is a way to access the Swing event-dispatching thread from outside event-handling or drawing code, is using *SwingUtilities' invokeLater()* and *invokeAndWait()* methods.
- A signed applet can become a trusted applet, which can work outside the sandbox.
- In Java typically memory leak occurs when an object of longer life cycle has a reference to objects of a short life cycle.
- You can improve performance in Java by :
  1. Pooling your valuable resources like threads, database and socket connections.
  2. Optimizing your I/O operations.
  3. Minimising network overheads, calls to *Date*, *Calendar* related classes, use of “**casting**” or runtime type checking like “**instanceof**” in frequently executed methods/loops, JNI calls, etc
  4. Managing your objects efficiently by caching or recycling them without having to rely on garbage collection.
  5. Using a *StringBuffer* as opposed to *String* and *ArrayList* or *HashMap* as oppose to *Vector* or *Hashtable*
  6. Applying multi-threading where applicable. **Factory: Abstraction or interface, to let sub class or the implementing class, to decide, which class is to be instantiated, or which method is to be called**
  7. Minimise any potential memory leaks.
- Finally, very briefly familiarise yourself with some of the key **design patterns** like:

**Prototype:**

Cloning of an object  
to reduce the cost of creation.

E.g. method overriding

**Singleton:**

Only one instance  
of the object exists  
at anytime

1. **Decorator design pattern:** used by Java I/O API. A popular design pattern.
2. **Reactor design pattern/Observer design pattern:** used by Java NIO API.
3. **Visitor design pattern:** to avoid instanceof and typecast constructs.
4. **Factory method/abstract factory design pattern:** popular pattern, which gets frequently asked in interviews.
5. **Singleton pattern:** popular pattern, which gets frequently asked in interviews.
6. **Composite design pattern:** used by GUI components and also a popular design pattern
7. **MVC design pattern/architecture:** used by Swing components and also a popular pattern.
8. **Command pattern:** used by Swing action architecture and also a popular design pattern.
9. **Strategy design pattern:** A popular design pattern used by AWT layout managers.

Refer Q11 in “How would you go about...” section for a detailed discussion and code samples on GOF (Gang of Four) design patterns.

**Recommended reading on design patterns:**

- The famous Gang of Four book: Design Patterns, Eric Gamma, Richard Helm, Ralph Johnson, and John Vlissides (Addiso-Wesley Publishing Co., 1995; ISBN: 0201633612).

**Tips:**

- Try to find out the needs of the project in which you will be working and the needs of the people within the project.
- 80% of the interview questions are based on your own resume.
- Where possible briefly demonstrate how you applied your skills/knowledge in the key areas [design concepts, transactional issues, performance issues, memory leaks etc] as described in this book. Find the right time to raise questions and answer those questions to show your strength.
- Be honest to answer technical questions, you are not expected to remember everything (for example you might know a few design patterns but not all of them etc). If you have not used a design pattern in question, request the interviewer, if you could describe a different design pattern.
- Do not be critical, focus on what you can do. Also try to be humorous to show your smartness.
- Do not act superior.

## SECTION TWO

### Enterprise Java – Interview questions & answers

K  
E  
Y  
A  
R  
E  
A  
S

- Specification Fundamentals **SF**
- Design Concepts **DC**
- Design Patterns **DP**
- Concurrency Issues **CI**
- Performance Issues **PI**
- Memory Issues **MI**
- Exception Handling **EH**
- Transactional Issues **TI**
- Security **SE**
- Scalability Issues **SI**
- Best Practices **BP**
- Coding<sup>1</sup> **CO**

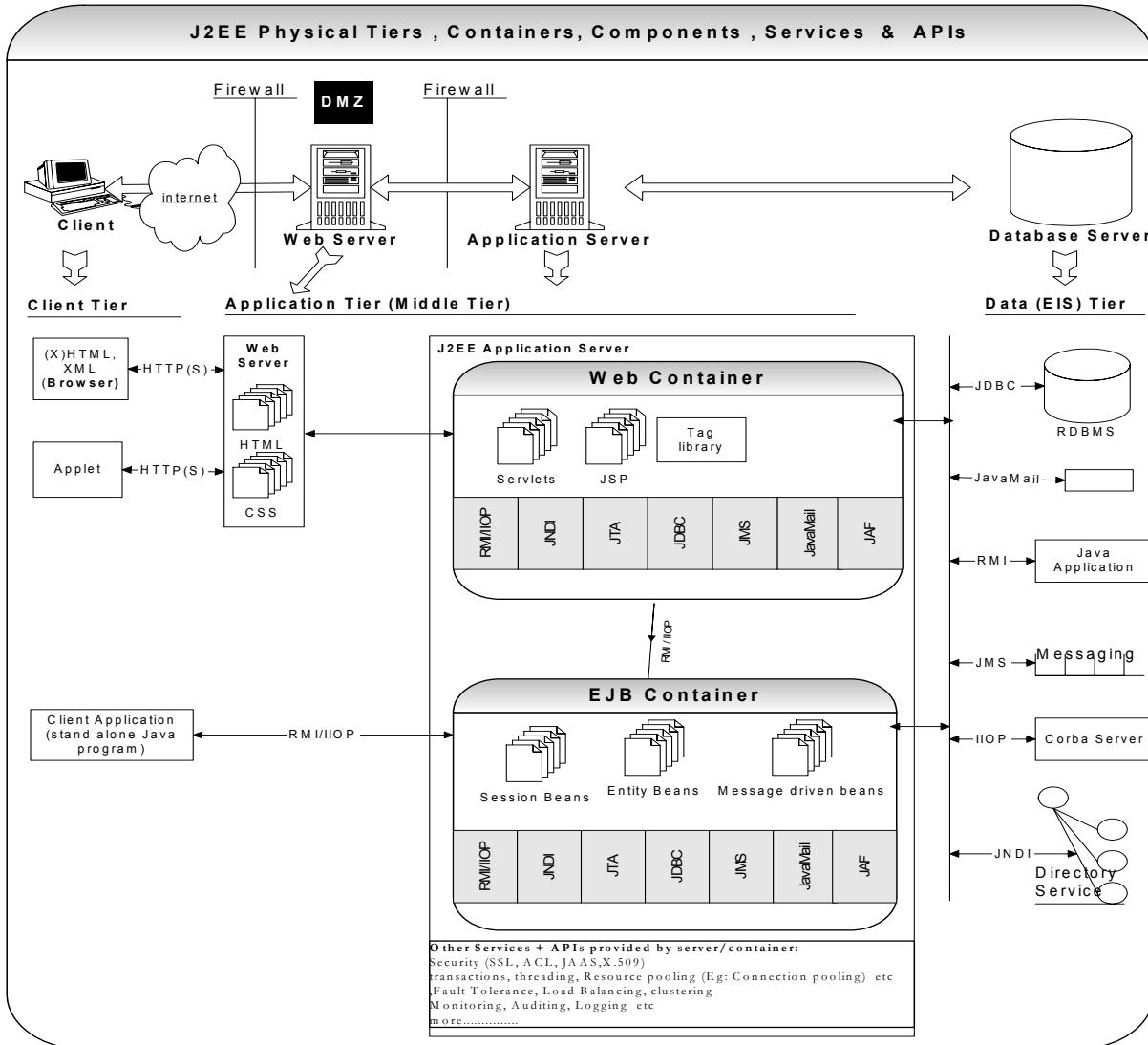
**Popular Questions:** Q02, Q03, Q10, Q16, Q19, Q20, Q24, Q25, Q30, Q31, Q36, Q39, Q40, Q45, Q46, Q48, Q49, Q53, Q58, Q63, Q64, Q65, Q66, Q71, Q72, Q73, Q76, Q77, Q78, Q79, Q83, Q84, Q85, Q86, Q87, Q89, Q90, Q91, Q93, Q96, Q97, Q98, Q100, Q102, Q106, Q107, Q110, Q123, Q124, Q125, Q129, **Q131**, Q136.

<sup>1</sup> Unlike other key areas, the **CO** is not always shown against the question but shown above the actual subsection of relevance within a question.

### Enterprise - J2EE

**Q 01:** What is J2EE? What are J2EE components and services? [SF](#)

**A 01:** J2EE (**J**ava **2** **E**nterprise **E**dition) is an environment for developing and deploying enterprise applications. The J2EE platform consists of J2EE components, services, Application Programming Interfaces (APIs) and protocols that provide the functionality for developing multi-tiered and distributed Web based applications.



A **J2EE component** is a self-contained functional software unit that is assembled into a J2EE application with its related classes and files and communicates with other components. The J2EE specification defines the following J2EE components:

Component type	Components	Packaged as
Applet	applets	JAR (Java ARchive)
Application client	Client side Java codes.	JAR (Java ARchive)
Web component	JSP, Servlet	WAR (Web ARchive)
Enterprise JavaBeans	Session beans, Entity beans, Message driven beans	JAR (EJB Archive)
Enterprise application	WAR, JAR, etc	EAR (Enterprise ARchive)
Resource adapters	Resource adapters	RAR (Resource Adapter ARchive)

**So what is the difference between a component and a service you may ask?** A component is an application level software unit as shown in the table above. All the J2EE components depend on the container for the system level support like transactions, security, pooling, life cycle management, threading etc. A service is a component

that can be used remotely through a remote interface either synchronously or asynchronously (e.g. Web service, messaging system, sockets, RPC etc).

**Containers** (Web & EJB containers) are the interface between a J2EE component and the low level platform specific functionality that supports J2EE components. Before a Web, enterprise bean (EJB), or application client component can be executed, it must be assembled into a J2EE module (jar, war, and/or ear) and deployed into its container.

A J2EE server provides **system level support services** such as security, transaction management, JNDI (Java Naming and Directory Interface) lookups, remote access etc. **J2EE architecture provides configurable and non-configurable services**. The configurable service enables the J2EE components within the same J2EE application to behave differently based on where they are deployed. For example the security settings can be different for the same J2EE application in two different production environments. The non-configurable services include enterprise bean (EJB) and servlet life cycle management, resource pooling etc.

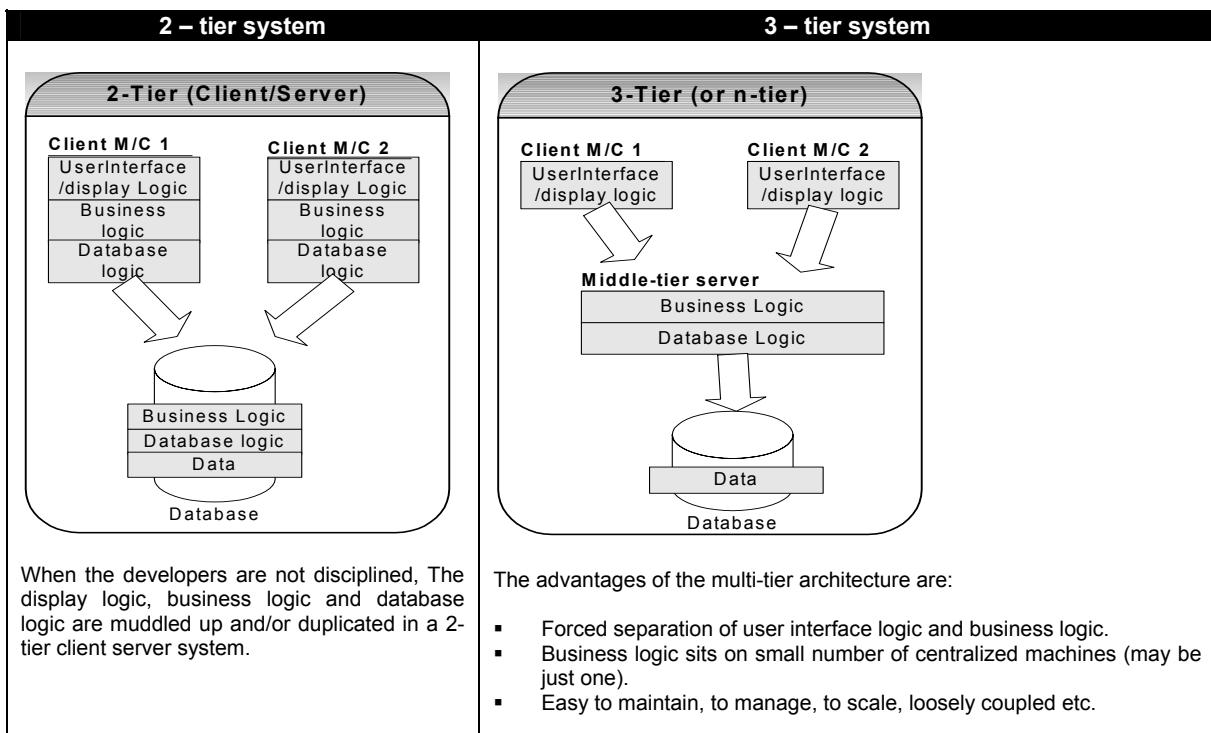
**Protocols** are used for access to Internet services. J2EE platform supports HTTP (HyperText Transfer Protocol), TCP/IP (Transmission Control Protocol / Internet Protocol), RMI (Remote Method Invocation), SOAP (Simple Object Access Protocol) and SSL (Secured Socket Layer) protocol.

The J2EE API can be summarised as follows:

J2EE technology category	API (Application Program Interface)
Component model technology	Java Servlet, JavaServer Pages(JSP), Enterprise JavaBeans(EJB).
Web services technology	JAXP (Java API for XML Processing), JAXR (Java API for XML Registries), SAAJ (SOAP with attachment API for Java), JAX-RPC (Java API for XML-based RPC), JAX-WS (Java API for XML-based Web Services).
Other	JDBC (Java Database Connectivity), JNDI (Java Naming and Directory Interface), JMS (Java Messaging Service), JCA (J2EE Connector Architecture), JTA (Java Transaction API), JavaMail, JAF (JavaBeans Activation Framework – used by JavaMail), JAAS (Java Authentication and Authorization Service), JMX (Java Management eXtensions).

**Q 02:** Explain the J2EE 3-tier or n-tier architecture? **SF DC**

**A 02:** This is a very commonly asked question. Be prepared to draw some diagrams on the board. The J2EE platform is a multi-tiered system. A tier is a logical or functional partitioning of a system.



Each tier is assigned a unique responsibility in a 3-tier system. Each tier is logically separated and loosely coupled from each other, and may be distributed.

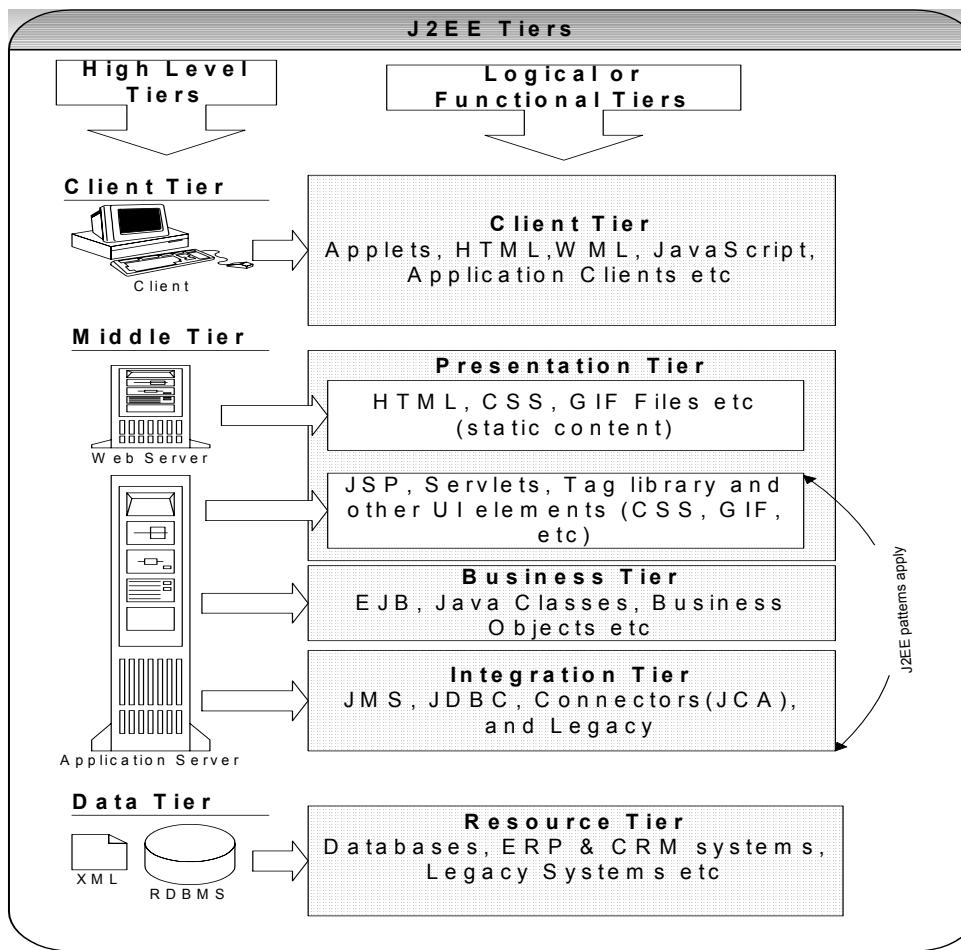
**Client tier** represents Web browser, a Java or other application, Applet, WAP phone etc. The client tier makes requests to the Web server who will be serving the request by either returning static content if it is present in the Web server or forwards the request to either Servlet or JSP in the application server for either static or dynamic content.

**Presentation tier** encapsulates the presentation logic required to serve clients. A Servlet or JSP in the presentation tier intercepts client requests, manages logons, sessions, accesses the business services, and finally constructs a response, which gets delivered to client.

**Business tier** provides the business services. This tier contains the business logic and the business data. All the business logic is centralised into this tier as opposed to 2-tier systems where the business logic is scattered between the front end and the backend. The benefit of having a centralised business tier is that same business logic can support different types of clients like browser, WAP, other stand-alone applications etc.

**Integration tier** is responsible for communicating with external resources such as databases, legacy systems, ERP systems, messaging systems like MQSeries etc. The components in this tier use JDBC, JMS, J2EE Connector Architecture (JCA) and some proprietary middleware to access the resource tier.

**Resource tier** is the external resource such as a database, ERP system, Mainframe system etc responsible for storing the data. This tier is also known as Data Tier or EIS (Enterprise Information System) Tier.



**Note:** On a high level J2EE can be construed as a **3-tier** system consisting of **Client Tier**, **Middle Tier** (or Application Tier) and **Data Tier**. But logically or functionally J2EE is a multi-tier (or n-tier) platform.

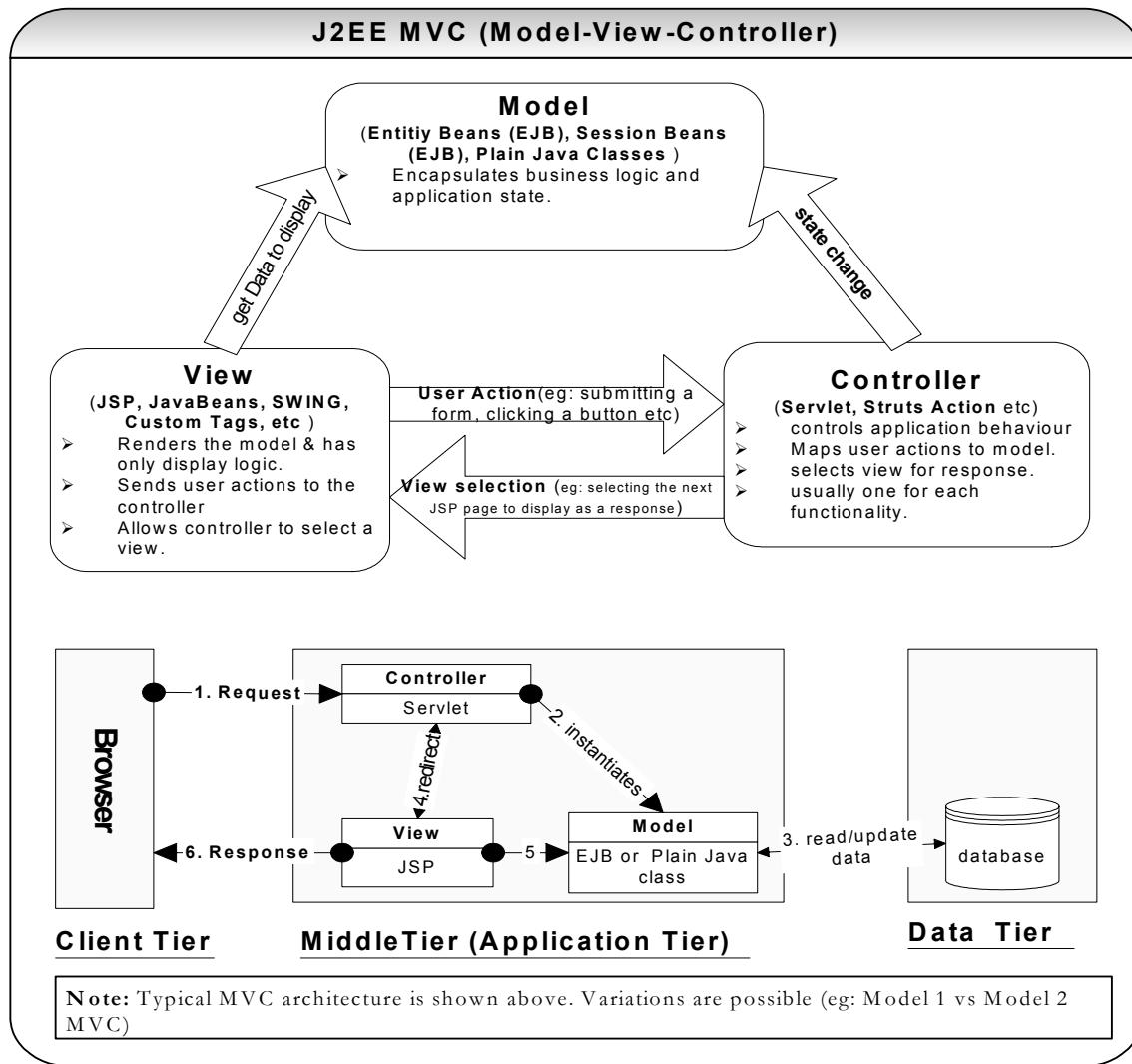
**The advantages of a 3-tiered or n-tiered application:** 3-tier or multi-tier architectures force separation among presentation logic, business logic and database logic. Let us look at some of the key benefits:

- **Manageability:** Each tier can be monitored, tuned and upgraded independently and different people can have clearly defined responsibilities.

- **Scalability:** More hardware can be added and allows clustering (i.e. horizontal scaling).
- **Maintainability:** Changes and upgrades can be performed without affecting other components.
- **Availability:** Clustering and load balancing can provide availability.
- **Extensibility:** Additional features can be easily added.

**Q 03:** Explain MVC architecture relating to J2EE? **DC DP**

**A 03:** This is also a very popular interview question. MVC stands for Model-View-Controller architecture. It divides the functionality of displaying and maintaining of the data to minimise the degree of coupling (i.e. promotes loose coupling) between components.



A **model** represents the **core business logic** and **state**. A model commonly maps to data in the database and will also contain core business logic.

A **View** renders the contents of a model. A view accesses the data from the model and adds **display logic** to present the data.

A **Controller** acts as the **glue between a model and a view**. A controller delegates the request to the model for application logic and state and also centralises the logic for dispatching the request to the next view based on the input parameters from the client and the application state. A controller also decouples JSP pages and the Servlet by handling the view selection.

**Q 04:** How to package a module, which is, shared by both the WEB and the EJB modules? **SF**

**A 04:** Package the modules shared by both WEB and EJB modules as dependency jar files. Define the **Class-Path:** property in the **MANIFEST.MF** file in the EJB jar and the Web war files to refer to the shared modules. [Refer Q7 in Enterprise section for diagram: *J2EE deployment structure*].

The **MANIFEST.MF** files in the EJB jar and WEB war modules should look like:

```
Manifest-Version: 1.0
Created-By: Apache Ant 1.5
Class-Path: myAppsUtil.jar
```

**Q 05:** Why use design patterns in a J2EE application? **DP**

**A 05:**

- **They have been proven.** Patterns reflect the experience and knowledge of developers who have successfully used these patterns in their own work. It lets you leverage the collective experience of the development community.

**Example** Session facade and value object patterns evolved from performance problems experienced due to multiple network calls to the EJB tier from the WEB tier. Fast lane reader and Data Access Object patterns exist for improving database access performance. The flyweight pattern improves application performance through object reuse (which minimises the overhead such as memory allocation, garbage collection etc).

- **They provide common vocabulary.** Patterns provide software designers with a common vocabulary. Ideas can be conveyed to developers using this common vocabulary and format.

**Example** Should we use a Data Access Object (DAO)? How about using a Business Delegate? Should we use Value Objects to reduce network overhead? Etc.

**Q 06:** What is the difference between a Web server and an application server? **SF**

**A 06:**

Web Server	Application Server
Supports HTTP protocol. When the Web server receives an HTTP request, it responds with an HTTP response, such as sending back an HTML page (static content) or delegates the dynamic response generation to some other program such as CGI scripts or Servlets or JSPs in the application server.	Exposes <b>business logic</b> and <b>dynamic content</b> to the client through various protocols such as HTTP, TCP/IP, IIOP, JRMP etc.
Uses various scalability and fault-tolerance techniques.	<p>Uses various scalability and fault-tolerance techniques. In addition provides resource pooling, component life cycle management, transaction management, messaging, security etc.</p> <p>Provides services for components like Web container for servlet components and EJB container for EJB components.</p>

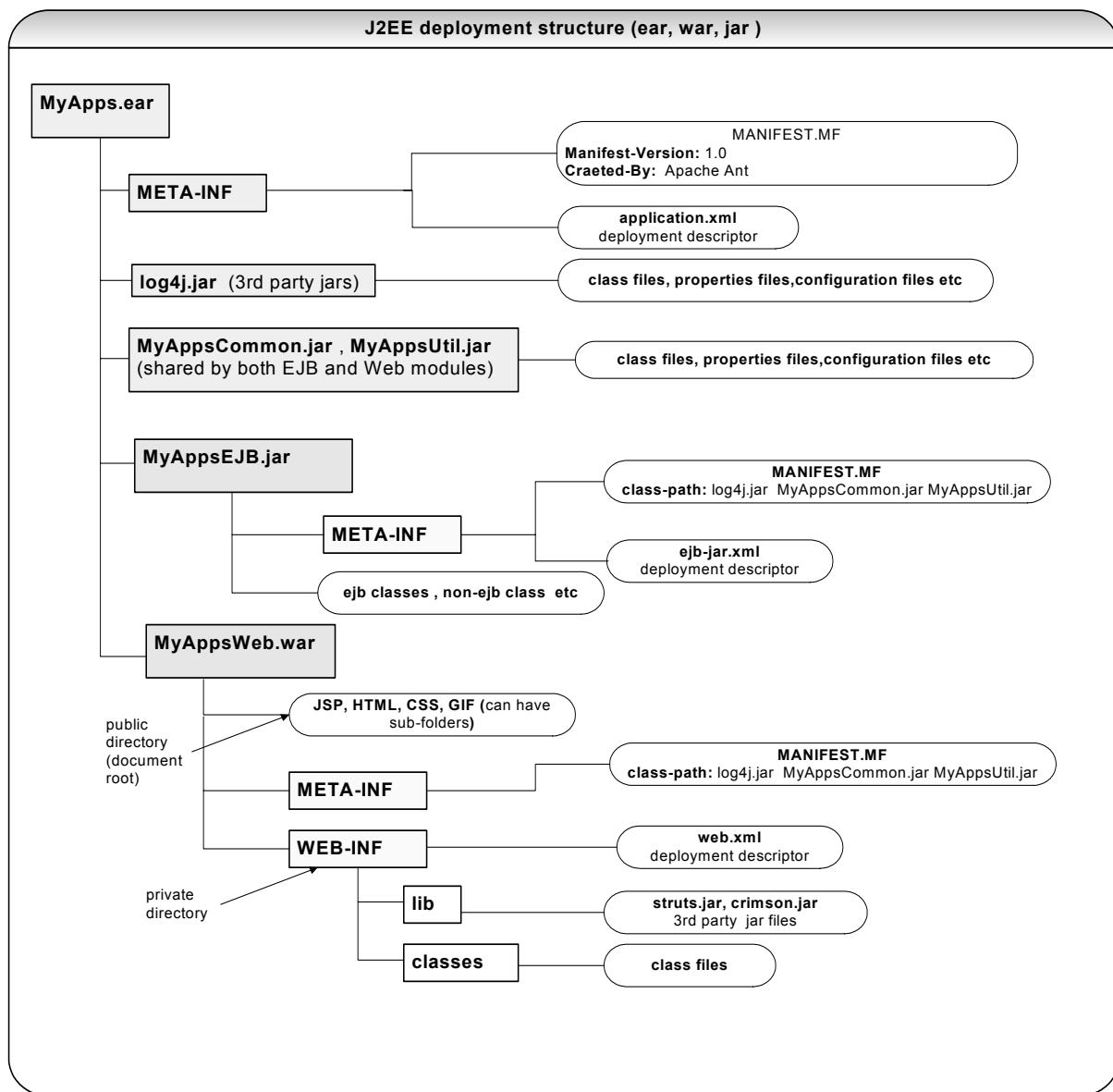
With the advent of XML Web services the line between application servers and Web servers is not clear-cut. By passing XML documents between request and response the Web server can behave like an application server.

**Q 07:** What are ear, war and jar files? What are J2EE Deployment Descriptors? **SF**

**A 07:** ear, war and jar are standard application deployment archive files. Since they are a standard, any application server (at least in theory) will know how to unpack and deploy them.

An EAR file is a standard JAR file with an “.ear” extension, named from Enterprise ARchive file. A J2EE application with all of its modules is delivered in EAR file. JAR files can't have other JAR files. But EAR and WAR (Web ARchive) files can have JAR files.

An EAR file contains all the JARs and WARs belonging to an application. JAR files contain the EJB classes and **WAR** files contain the Web components (JSPs, static content (HTML, CSS, GIF etc), Servlets etc.). The J2EE application client's class files are also stored in a JAR file. EARs, JARs, and WARs all contain an XML-based deployment descriptor.



### Deployment Descriptors

A deployment descriptor is an XML based text file with a “.xml” extension that describes a component's deployment settings. A J2EE application and each of its modules has its own deployment descriptor. Pay attention to elements marked in bold in the sample deployment descriptor files shown below.

- **application.xml:** is a standard J2EE deployment descriptor, which includes the following structural information: EJB jar modules, WEB war modules, <security-role> etc. Also since EJB jar modules are packaged as jars the same way dependency libraries like log4j.jar, commonUtil.jar etc are packaged, the application.xml descriptor will distinguish between these two jar files by explicitly specifying the EJB jar modules.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE application PUBLIC "-//Sun Microsystems, Inc.//DTD J2EE Application 1.2//EN"
 "http://java.sun.com/j2ee/dtds/application_1_2.dtd">
<application id="Application_ID">
 <display-name>MyApps</display-name>
 <module id="EjbModule_1">
 <ejb>MyAppsEJB.jar</ejb>
 </module>

 <module id="WebModule_1">
 <web>

```

```

<web-uri>MyAppsWeb.war</web-uri>
<context-root>myAppsWeb</context-root>
</web>
</module>

<security-role id="SecurityRole_1">
 <description>Management position</description>
 <role-name>manager</role-name>
</security-role>
</application>

```

- **ejb-jar.xml:** is a standard deployment descriptor for an EJB module.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 1.1//EN"
 "http://java.sun.com/j2ee/dtds/ejb-jar_1_1.dtd">
<ejb-jar id="ejb-jar_ID">
 <display-name>MyAppsEJB</display-name>

 <enterprise-beans>
 <session id="ContentService">
 <ejb-name>ContentService</ejb-name>
 <home>ejb.ContentServiceHome</home>
 <remote>ejb.ContentService</remote>
 <ejb-class>ejb.ContentServiceBean</ejb-class>
 <session-type>Stateless</session-type>
 <transaction-type>Bean</transaction-type>
 </session>

 <entity>
 <ejb-name>Bid</ejb-name>
 <home>ejb.BidHome</home>
 <remote>ejb.Bid</remote>
 <ejb-class>ejb.BidBean</ejb-class>
 <persistence-type>Container</persistence-type>
 <prim-key-class>ejb.BidPK</prim-key-class>
 <reentrant>False</reentrant>
 <cmp-field><field-name>bid</field-name></cmp-field>
 <cmp-field><field-name>bidding</field-name></cmp-field>
 <cmp-field><field-name>bidDate</field-name></cmp-field>
 <cmp-field><field-name>id</field-name></cmp-field>
 </entity>
 </enterprise-beans>

 <!-- OPTIONAL -->

 <assembly-descriptor>
 <!-- OPTIONAL, can be many -->
 <security-role>
 <description>
 Employee is allowed to ...
 </description>
 <role-name>employee</role-name>
 </security-role>

 <!-- OPTIONAL. Can be many -->
 <method-permission>
 <!-- Define role name in "security-role" -->
 <!-- Must be one or more -->
 <role-name>employee</role-name>
 <!-- Must be one or more -->
 <method>
 <ejb-name>ContentService</ejb-name>
 <!-- * = all methods -->
 <method-name>*</method-name>
 </method>

 <method>
 <ejb-name>Bid</ejb-name>
 <method-name>findByPrimaryKey</method-name>
 </method>
 </method-permission>
 </assembly-descriptor>

```

```

<!-- OPTIONAL, can be many. How the container is to manage
transactions when calling an EJB's business methods -->

<container-transaction>
 <!-- Can specify many methods at once here -->
 <method>
 <ejb-name>Bid</ejb-name>
 <method-name>*</method-name>
 </method>
 <!-- NotSupported|Supports|Required|RequiresNew|Mandatory|Never -->
 <trans-attribute>Required</trans-attribute>
</container-transaction>
</assembly-descriptor>

</ejb-jar>

```

- **web.xml:** is a standard deployment descriptor for a WEB module.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
 "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
 <display-name>myWebApplication</display-name>
 <context-param>
 <param-name>GlobalContext.ClassName</param-name>
 <param-value>web.GlobalContext</param-value>
 </context-param>

 <servlet>
 <servlet-name>MyWebController</servlet-name>
 <servlet-class>web.MyWebController</servlet-class>
 <init-param>
 <param-name>config</param-name>
 <param-value>/WEB-INF/config/myConfig.xml</param-value>
 </init-param>
 <load-on-startup>1</load-on-startup>
 </servlet>

 <servlet-mapping>
 <servlet-name>MyWebController</servlet-name>
 <url-pattern>/execute/*</url-pattern>
 </servlet-mapping>

 <error-page>
 <error-code>400</error-code>
 <location>/WEB-INF/jsp/errors/myError.jsp</location>
 </error-page>

 <taglib>
 <taglib-uri>/WEB-INF/struts-bean.tld</taglib-uri>
 <taglib-location>/WEB-INF/lib/taglib/struts/struts-bean.tld</taglib-location>
 </taglib>

 <security-constraint>
 <web-resource-collection>
 <web-resource-name>Employer</web-resource-name>
 <description></description>
 <url-pattern>/execute/employ</url-pattern>
 <http-method>POST</http-method>
 <http-method>GET</http-method>
 <http-method>PUT</http-method>
 </web-resource-collection>
 <auth-constraint>
 <description></description>
 <role-name>advisor</role-name>
 </auth-constraint>
 </security-constraint>

 <login-config>
 <auth-method>FORM</auth-method>
 <realm-name>FBA</realm-name>
 <form-login-config>
 <form-login-page>/execute/MyLogon</form-login-page>

```

```

<form-error-page>/execute/MyError</form-error-page>
</form-login-config>
</login-config>

<security-role>
 <description>Advisor</description>
 <role-name>advisor</role-name>
</security-role>

</web-app>

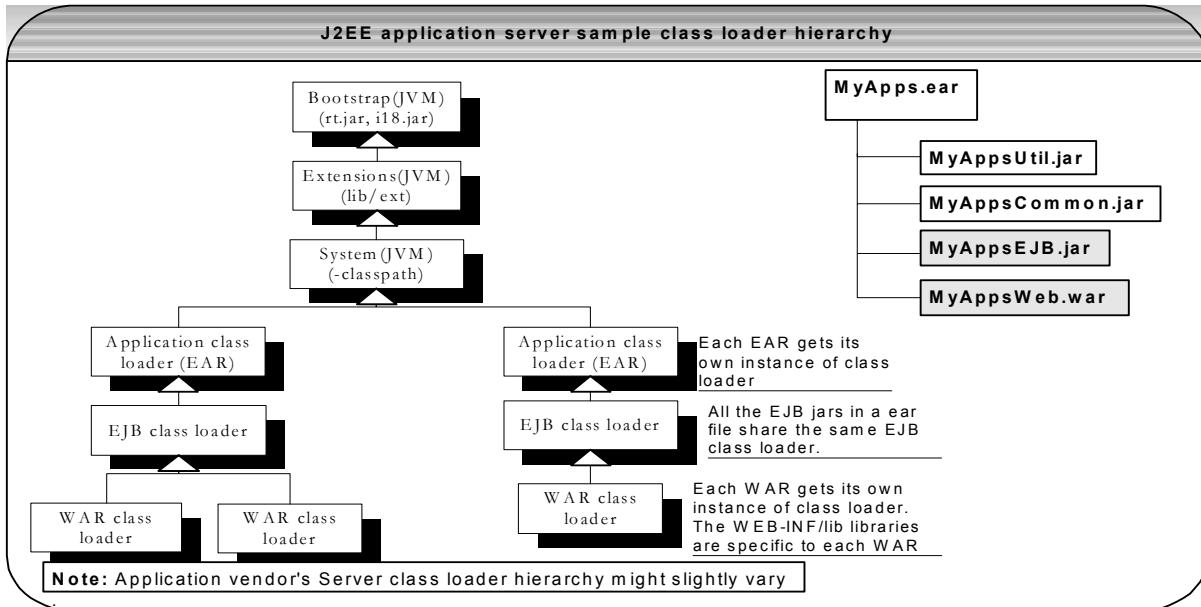
```

**Q 08:** Explain J2EE class loaders? **SF**

**A 08:** J2EE application server sample class loader hierarchy is shown below. (Also refer to **Q4** in Java section). As per the diagram the J2EE application specific class loaders are children of the “System –classpath” class loader. When the parent class loader is above the “System –Classpath” class loader in the hierarchy as shown in the diagram (i.e. bootstrap class loader or extensions class loader) then child class loaders implicitly have visibility to the classes loaded by its parents. When a parent class loader is below a “System -Classpath” class loader then the child class loaders will only have visibility into the classes loaded by its parents **only if they are explicitly specified in a manifest file** (MANIFEST.MF) of the child class loader.

**Example** As per the diagram, if the EJB module *MyAppsEJB.jar* wants to refer to *MyAppsCommon.jar* and *MyAppsUtil.jar* we need to add the following entry in the *MyAppsEJB.jar*’s manifest file MANIFEST.MF.

**class-path:** *MyAppsCommon.jar* *MyAppsUtil.jar*



This is because the application (EAR) class loader loads the *MyAppsCommon.jar* and *MyAppsUtil.jar*. The EJB class loader loads the *MyAppsEJB.jar*, which is the child class loader of the application class loader. The WAR class loader loads the *MyAppsWeb.war*.

Every J2EE application or EAR gets its own instance of the application class loader. This class loader is responsible for loading all the dependency jar files, which are shared by both WEB and EJB modules. For example third party libraries like log4j, utility classes, shared classes or common classes (Exception thrown by an EJB module should be caught by a WEB module) etc.

The key difference between the EJB and WAR class loader is that all the EJB jars in the application **share the same EJB class loader** whereas WAR files get their own class loader. This is because the EJBs have inherent relationship between one another (ie EJB-EJB communication between EJBs in different applications but hosted on the same JVM) but the Web modules do not. Every WAR file should be able to have its own WEB-INF/lib third party libraries and need to be able to load its own version of converted logon.jsp Servlet so each WEB module is isolated in its own class loader.

So if two different WEB modules want to use two different versions of the same EJB then we need to have two different ear files. As was discussed in the Q4 in Java section the class loaders use a **delegation model** where the child class loaders delegate the loading up the hierarchy to their parent before trying to load it itself only if the parent can't load it. But with regards to WAR class loaders, some application servers provide a setting to turn this behaviour off (DelegationMode=false). This delegation mode is recommended in the Servlet 2.3 specification.

As a general rule **classes should not be deployed higher in the hierarchy than they are supposed to exist**. This is because if you move one class up the hierarchy then you will have to move other classes up the hierarchy as well. This is because classes loaded by the parent class loader can't see the classes loaded by its child class loaders (**uni-directional bottom-up visibility**).

### Enterprise - Servlet

**Q 09:** What is the difference between CGI and Servlet? **SF**

**Q 09:**

Traditional CGI (Common Gateway Interface)	Java Servlet
Traditional CGI creates a heavy weight process to handle each http request. N number of copies of the same traditional CGI programs is copied into memory to serve N number of requests.	Spawns a lightweight Java thread to handle each http request. Single copy of a type of servlet but N number of threads (thread sizes can be configured in an application server).

In the Model 2 MVC architecture, servlets process requests and select JSP views. So servlets act as controller. Servlets intercept the incoming HTTP requests from the client (browser) and then dispatch the request to the business logic model (e.g. EJB, POJO - Plain Old Java Object, JavaBeans etc). Then select the next JSP view for display and deliver the view to client as the presentation (response). It is the best practice to use Web tier UI frameworks like Struts, JavaServer Faces etc, which uses proven and tested design patterns.

**Q 10:** HTTP is a stateless protocol, so how do you maintain state? How do you store user data between requests? **SF**

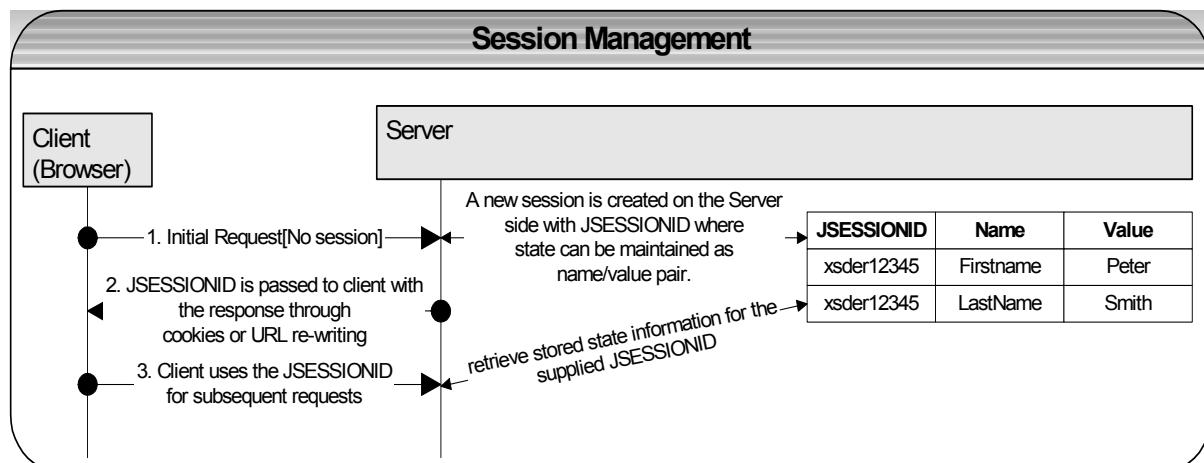


**A 10:** This is a commonly asked question as well. You can retain the state information between different page requests as follows:

**HTTP Sessions** are the recommended approach. A session identifies the requests that originate from the same browser during the period of conversation. All the servlets can share the same session. The JSESSIONID is generated by the server and can be passed to client through cookies, URL re-writing (if cookies are turned off) or built-in SSL mechanism. Care should be taken to **minimize size of objects stored in session and objects stored in session should be serializable**. In a Java servlet the session can be obtained as follows: **CO**

```
HttpSession session = request.getSession(); //returns current session or a new session
```

Sessions can be timed out (configured in web.xml) or manually invalidated.



**Hidden Fields** on the pages can maintain state and they are not visible on the browser. The server treats both hidden and non-hidden fields the same way.

```
<INPUT type="hidden" name="Firstname" value="Peter">
<INPUT type="hidden" name="Lastname" value="Smith">
```

The disadvantage of hidden fields is that they may expose sensitive or private information to others.

**URL re-writing** will append the state information as a query string to the URL. This should not be used to maintain private or sensitive information.

```
Http://MyServer:8080/MyServlet?Firstname=Peter&Lastname=Smith
```

**Cookies:** A cookie is a piece of text that a Web server can store on a user's hard disk. Cookies allow a website to store information on a user's machine and later retrieve it. These pieces of information are stored as name-value pairs. The cookie data moves in the following manner:

- ❖ If you type the URL of a website into your browser, your browser sends the request to the Web server. When the browser does this it looks on your machine for a cookie file that URL has set. If it finds it, your browser will send all of the name-value pairs along with the URL. If it does not find a cookie file, it sends no cookie data.
- ❖ The URL's Web server receives the cookie data and requests for a page. If name-value pairs are received, the server can use them. If no name-value pairs are received, the server can create a new ID and then sends name-value pairs to your machine in the header for the Web page it sends. Your machine stores the name value pairs on your hard disk.

Cookies can be used to determine how many visitors visit your site. It can also determine how many are new versus repeated visitors. The way it does this is by using a database. The first time a visitor arrives, the site creates a new ID in the database and sends the ID as a cookie. The next time the same user comes back, the site can increment a counter associated with that ID in the database and know how many times that visitor returns. The sites can also store user preferences so that site can look different for each visitor.

#### Which mechanism to choose?

Session mechanism	Description
HttpSession	<ul style="list-style-type: none"> <li>▪ There is no limit on the size of the session data kept.</li> <li>▪ The performance is good.</li> <li>▪ This is the preferred way of maintaining state. If we use the HTTP session with the application server's persistence mechanism (server converts the session object into BLOB type and stores it in the Database) then the performance will be moderate to poor.</li> </ul> <p><b>Note:</b> When using HttpSession mechanism you need to take care of the following points:</p> <ul style="list-style-type: none"> <li>▪ Remove session explicitly when you no longer require it.</li> <li>▪ Set the session timeout value.</li> <li>▪ Your application server may serialize session objects after crossing a certain memory limit. This is expensive and affects performance. So decide carefully what you want to store in a session.</li> </ul>
Hidden fields	<ul style="list-style-type: none"> <li>▪ There is no limit on size of the session data.</li> <li>▪ May expose sensitive or private information to others (So not good for sensitive information).</li> <li>▪ The performance is moderate.</li> </ul>
URL rewriting	<ul style="list-style-type: none"> <li>▪ There is a limit on the size of the session data.</li> <li>▪ Should not be used for sensitive or private information.</li> <li>▪ The performance is moderate.</li> </ul>
Cookies	<ul style="list-style-type: none"> <li>▪ There is a limit for cookie size.</li> <li>▪ The browser may turn off cookies.</li> <li>▪ The performance is moderate.</li> </ul> <p>The benefit of the cookies is that state information can be stored regardless of which server the client talks to and even if all servers go down. Also, if required, state information can be retained across sessions.</p>

**Q 11:** Explain the life cycle methods of a servlet? **SF**

**A 11:** The Web container is responsible for managing the servlet's life cycle. The Web container creates an instance of the servlet and then the container calls the init() method. At the completion of the init() method the servlet is in

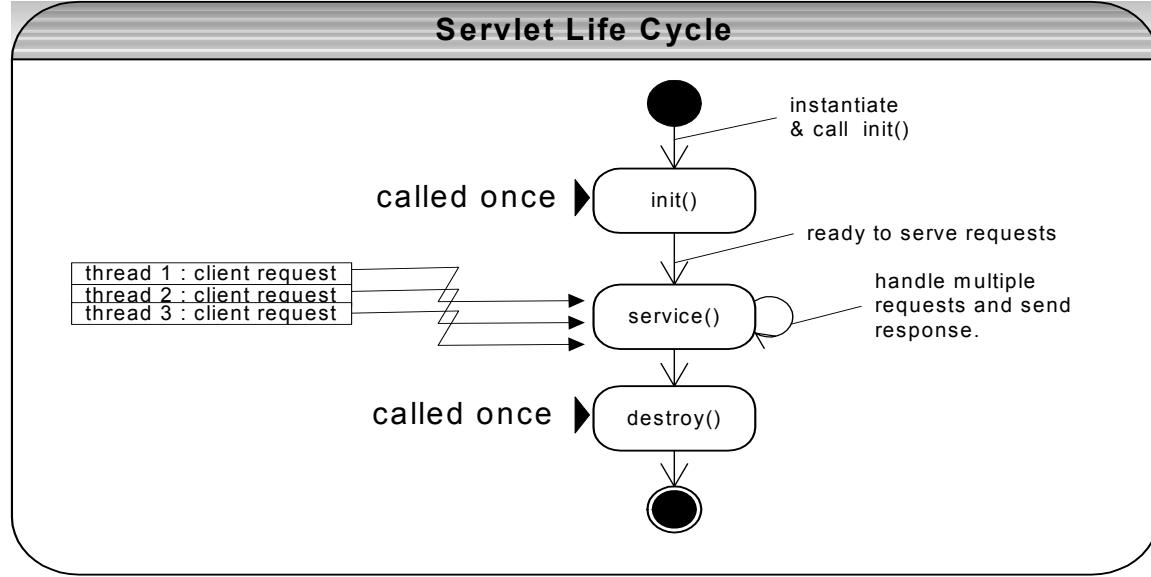
In case of thread pool, a group of fixed size threads are created. A thread from the thread pool is pulled out and assigned a job by the service provider. After completion of the job, thread is contained in the thread pool again. Advantage of Java Thread Pool is, Better performance It saves time because there is no need to create new thread. Real time usage is, It is used in Servlet and JSP where container creates a thread pool to process the request.

ready state to service requests from clients. The container calls the servlet's service() method for handling each request by spawning a new thread for each request from the Web container's thread pool [It is also possible to have a single threaded Servlet, refer **Q16** in Enterprise section]. Before destroying the instance the container will call the destroy() method. After destroy() the servlet becomes the potential candidate for garbage collection.

**Note on servlet reloading:**

Most servers can reload a servlet after its class file has been modified provided the servlets are deployed to **\$server\_root/servlets** directory. This is achieved with the help of a custom class loader. This feature is handy for development and test phases. This is not recommended for production since it can degrade performance because of timestamp comparison for each request to determine if a class file has changed. So for production it is recommended to move the servlet to server's class path ie **\$server\_root/classes**.

When a server dispatches a request to a servlet, the server first checks if the servlet's class file has changed on disk. If it has changed, the server abandons the class loader used to load the old version and creates a new instance of the custom class loader to load the new version. Old servlet versions can stay in memory indefinitely (so the effect is the other classes can still hold references to the old servlet instances, causing odd side effects, but the old versions are not used to handle any more requests. Servlet reloading is not performed for classes found in the server's classpath because the core, primordial class loader, loads those classes. These classes are loaded once and retained in memory even when their class files change.



**Q 12:** Explain the directory structure of a WEB application? **SF SE**

**A 12:** Refer **Q7** in Enterprise section for diagram: *J2EE deployment structure* and explanation in this section where *MyAppsWeb.war* is depicting the Web application directory structure. The directory structure of a Web application consists of two parts:

- A **public** resource directory (**document root**): The document root is where JSP pages, client-side classes and archives, and static Web resources are stored.
- A **private** directory called **WEB-INF**: which contains following files and directories:
  - **web.xml** : Web application deployment descriptor.
  - **\*.tld** : Tag library descriptor files.
  - **classes** : A directory that contains server side classes like servlets, utility classes, JavaBeans etc.
  - **lib** : A directory where JAR (archive files of tag libraries, utility libraries used by the server side classes) files are stored.

**Note:** JSP resources usually reside directly or under subdirectories of the **document root**, which **are directly accessible** to the user through the URL. If you want to protect your Web resources then hiding the JSP files behind the **WEB-INF** directory can protect the JSP files from direct access. Refer **Q35** in Enterprise section.

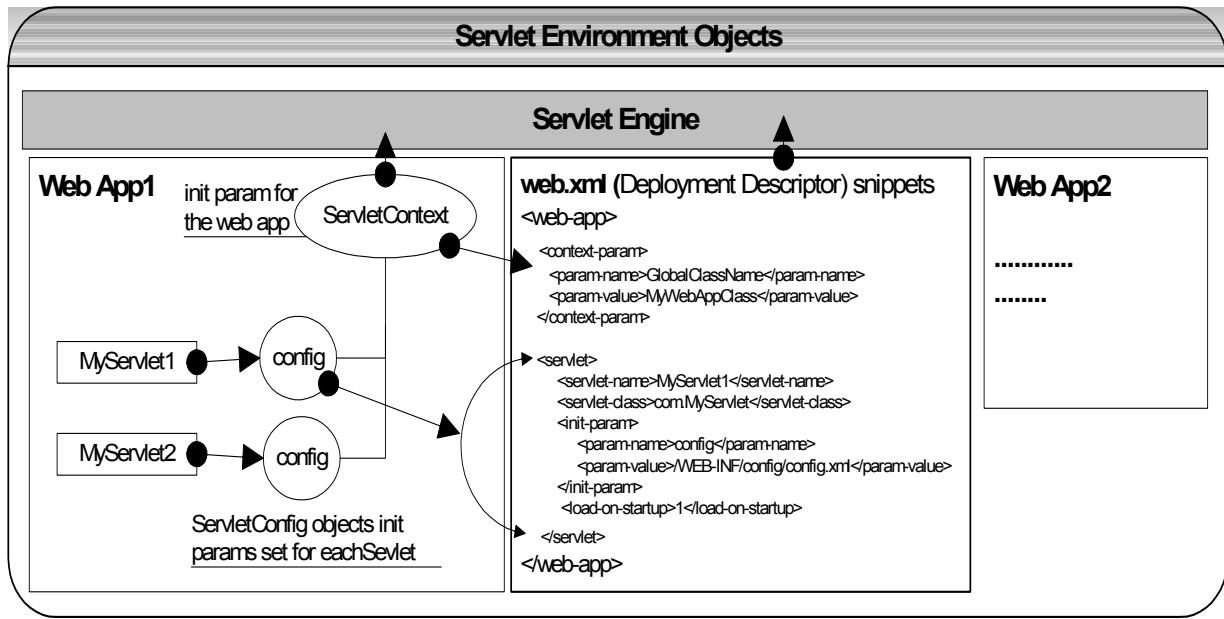
**Q 13:** What is the difference between doGet () and doPost () or GET and POST? **SF SE**

**A 13:**

<b>GET or doGet()</b>	<b>POST or doPost()</b>
The request parameters are transmitted as a query string appended to the request. Allows browser bookmarks but not appropriate for transmitting private or sensitive information. <code>http://MyServer/MyServlet?name=paul</code> This is a security risk.	The request parameters are passed with the body of the request. More secured.
GET was originally intended for static resource retrieval.	POST was intended for input data submits where the results are expected to change.
GET is not appropriate when large amounts of input data are being transferred.	

**Q 14:** What are the ServletContext and ServletConfig objects? What are Servlet environment objects? **SF****A 14:** The Servlet Engine uses both objects.

<b>ServletConfig</b>	<b>ServletContext</b>
The ServletConfig parameters are for <b>a particular Servlet</b> . The parameters are specified in the web.xml (ie deployment descriptor).	The ServletContext parameters are specified for the <b>entire Web application</b> . The parameters are specified in the web.xml (ie deployment descriptor).

**Q 15:** What is the difference between HttpServlet and GenericServlet? **SF****A 15:**

<b>GenericServlet</b>	<b>HttpServlet</b>
A GenericServlet has a service() method to handle requests.	The HttpServlet extends GenericServlet and adds support for HTTP protocol based methods like doGet(), doPost(), doHead() etc.
Protocol independent.	Protocol dependent.

**Q 16:** How do you make a Servlet thread safe? What do you need to be concerned about with storing data in Servlet instance fields? **C** **P** **BP****A 16:** As shown in the figure *Servlet Life Cycle* in **Q11** in Enterprise section, a typical (or default) Servlet life cycle creates a single instance of each servlet and creates multiple threads to handle the service() method. **The multi-threading aids efficiency but the servlet code must be coded in a thread safe manner.** The shared resources (e.g. instance variables, utility or helper objects etc) should be appropriately synchronized or should only use variables in a read-only manner. Having large chunks of code in synchronized blocks in your service methods can adversely affect performance and makes the code more complex.

Alternatively it is possible to have a **single threaded model of a servlet** by implementing the marker or null interface javax.servlet.SingleThreadedModel. The container will use one of the following approaches to ensure thread safety:

- **Instance pooling** where container maintains a pool of servlets.
- **Sequential processing** where new requests will wait while the current request is being processed.

**Best practice:** It is best practice to use multi-threading and stay away from the **single threaded model of the servlet** unless otherwise there is a compelling reason for it. Shared resources can be synchronized or used in read-only manner or shared values can be stored in a database table. The single threaded model can adversely affect performance.

---

**Q 17:** What is pre-initialization of a Servlet? **LF**

**A 17:** By default the container does not initialize the servlets as soon as it starts up. It initializes a servlet when it receives a request for the first time for that servlet. This is called **lazy loading**. The servlet deployment descriptor (web.xml) defines the <load-on-startup> element, which can be configured to make the servlet container load and initialize the servlet as soon as it starts up. The process of loading a servlet before any request comes in is called **pre-loading** or **pre-initializing** a servlet. We can also specify the order in which the servlets are initialized.

```
<load-on-startup>2</load-on-startup>
```

**Q 18:** What is a RequestDispatcher? What object do you use to forward a request? **LF CO**

**A 18:** A Servlet can obtain its **RequestDispatcher** object from its **ServletContext**.

```
//...inside the doGet() method
ServletContext sc = getServletContext();
RequestDispatcher rd = sc.getRequestDispatcher(url);

// forwards the control to another servlet or JSP to generate response. This method allows one servlet to do preliminary
//processing of a request and another resource to generate the response
rd.forward(request,response);
or
// includes the content of the resource such as Servlet, JSP, HTML, Images etc into the calling Servlet's response.
rd.include(request, response);
```

**Q 19:** What is the difference between forwarding a request and redirecting a request? **LF DC**

**A 19:** Both methods redirect you to a new resource like Servlet, JSP etc. But

redirecting - <b>sendRedirect()</b>	forward
Sends a header back to the browser, which contains the name of the resource to be redirected to. The browser will make a <b>fresh request from this header information</b> . Need to provide absolute URL path.	Forward action takes place within the server <b>without the knowledge of the browser</b> .
Has an overhead of extra remote trip but has the advantage of being able to refer to any resource on the same or different domain and also allows book marking of the page.	No extra network trip.

**Q 20:** What are the considerations for servlet clustering? **DC SI**

**A 20:** The clustering promotes high availability and scalability. The considerations for servlet clustering are:

- **Objects stored in a session should be serializable** to support in-memory replication of sessions. Also consider the overhead of serializing very large objects. Test the performance to make sure it is acceptable.
- **Design for idempotence**. Failure of a request or impatient users clicking again can result in duplicate requests being submitted. So the Servlets should be able to tolerate duplicate requests.
- **Avoid using instance and static variables in read and write mode** because different instances may exist on different JVMs. Any state should be held in an external resource such as a database.
- **Avoid storing values in a ServletContext**. A ServletContext is not serializable and also the different instances may exist in different JVMs.
- **Avoid using java.io.\* because the files may not exist on all backend machines**. Instead use getResourceAsStream().

**Q 21:** If an object is stored in a session and subsequently you change the state of the object, will this state change replicated to all the other distributed sessions in the cluster? **DC SI**

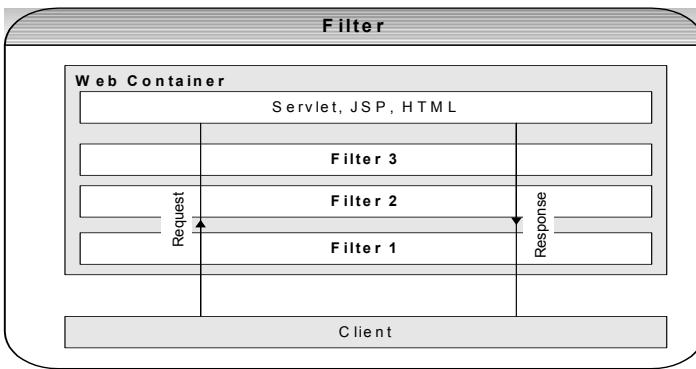
**A 21:** **No.** Session replication is the term that is used when your current service state is being replicated across multiple application instances. Session replication occurs when we replicate the information (ie **session attributes**) that are stored in your HttpSession. The container propagates the changes only when you call the **setAttribute(.....)** method. So mutating the objects in a session and then by-passing the **setAttribute(.....)** will not replicate the state change. **CO**

**Example** If you have an ArrayList in the session representing shopping cart objects and if you just call **getAttribute()** to retrieve the ArrayList and then add or change something without calling the **setAttribute(.....)** then the container may not know that you have added or changed something in the ArrayList. So the session will not be replicated.

**Q 22:** What is a filter, and how does it work? **LF DP**

**A 22:** A filter dynamically intercepts requests and responses to transform or use the information contained in the requests or responses but typically do not themselves create responses. Filters can also be used to transform the response from the Servlet or JSP before sending it back to client. Filters improve reusability by placing recurring tasks in the filter as a reusable unit.

A good way to think of Servlet filters is as a chain of steps that a request and response must go through before reaching a Servlet, JSP, or static resource such as an HTML page in a Web application.



The filters can be used for caching and compressing content, logging and auditing, image conversions (scaling up or down etc), authenticating incoming requests, XSL transformation of XML content, localization of the request and the response, site hit count etc. The filters are configured through the web.xml file as follows:

```
<web-app>
 <filter>
 <filter-name>HitCounterFilter</filter-name>
 <filter-class>myPkg.HitCounterFilter</filter-class>
 </filter>

 <filter-mapping>
 <filter-name>HitCounterFilter</filter-name>
 <url-pattern>/usersection/*</url-pattern>
 </filter-mapping>
 ...
</web-app>
```

The *HitCounterFilter* will intercept the requests from the URL pattern */usersection* followed by any resource name.

**Design Pattern:** Servlet filters use the slightly modified version of the **chain of responsibility** design pattern. Unlike the classic (only one object in the chain handle the request) chain of responsibility where filters allow multiple objects (filters) in a chain to handle the request. If you want to modify the request or the response in the chain you can use the **decorator pattern** (Refer Q11 in How would you go about... section).

**Q 23:** Explain declarative security for WEB applications? **SE**

### BEHAVIORAL Design Patterns :

**Chain of Responsibility:** Let more than one object participate in handling the request without the knowledge of each other. E.g. servlet chaining

**Observer:** One object changes state and all dependent objects are updated automatically. E.g. wait-notify

**A 23:** Servlet containers implement declarative security. The administration is done through the deployment descriptor web.xml file. With **declarative security** the Servlets and JSP pages will be free from any security aware code. You can protect your URLs through web.xml as shown below:

```

<web-app>
 <security-constraint>
 <web-resource-collection>
 <web-resource-name>PrivateAndSensitive</web-resource-name>
 <url-pattern>/private/*</url-pattern>
 </web-resource-collection>
 <auth-constraint>
 <role-name>executive</role-name>
 <role-name>admin</role-name>
 </auth-constraint>
 </security-constraint>

 <!-- form based authorization -->
 <login-config>
 <auth-method>FORM</auth-method>
 <form-login-config>
 <form-login-page>/login.jsp</form-login-page>
 <form-error-page>/error.jsp</form-error-page>
 </form-login-config>
 </login-config>
</web-app>

```

The user will be prompted for the configured login.jsp when restricted resources are accessed. The container also keeps track of which users have been previously authenticated.

**Benefits:** Very little coding is required and developers can concentrate on the application they are building and system administrators can administer the security settings without or with minimal developer intervention. Let's look at a sample programmatic security in a Web module like a servlet: **CO**

```

User user = new User();
Principal principal = request.getUserPrincipal();
if (request.isUserInRole("boss"))
 user.setRole(user.BOSS_ROLE);

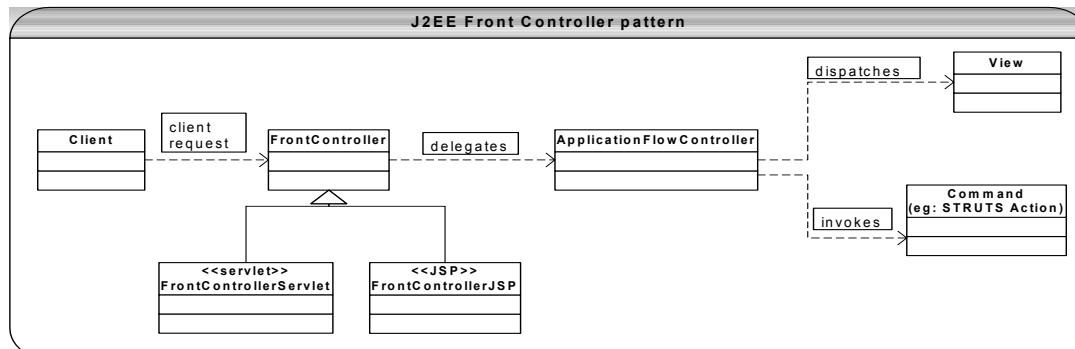
```

**Q 24:** Explain the **Front Controller** design pattern or explain J2EE design patterns? **DP**

**A 24: Problem:** A J2EE system requires a centralized access point for HTTP request handling to support the integration of system services like security, data validation etc, content retrieval, view management, and dispatching. When the user accesses the view directly without going through a centralized mechanism, two problems may occur:

- Each view is required to provide its own system services often resulting in **duplicate code**.
- View navigation is left to the views. This may result in shared code for view content and view navigation.
- Distributed control is **more difficult to maintain**, since changes will often need to be made in numerous places.

**Solution:** Generally you write specific servlets for specific request handling. These servlets are responsible for data validation, error handling, invoking business services and finally forwarding the request to a specific JSP view to display the results to the user.



The **Front Controller** suggests that we **only have one Servlet** (instead of having specific Servlet for each specific request) centralising the handling of all the requests and delegating the functions like validation, invoking business services etc to a command or a helper component. For example Struts framework uses the command design pattern to delegate the business services to an action class.

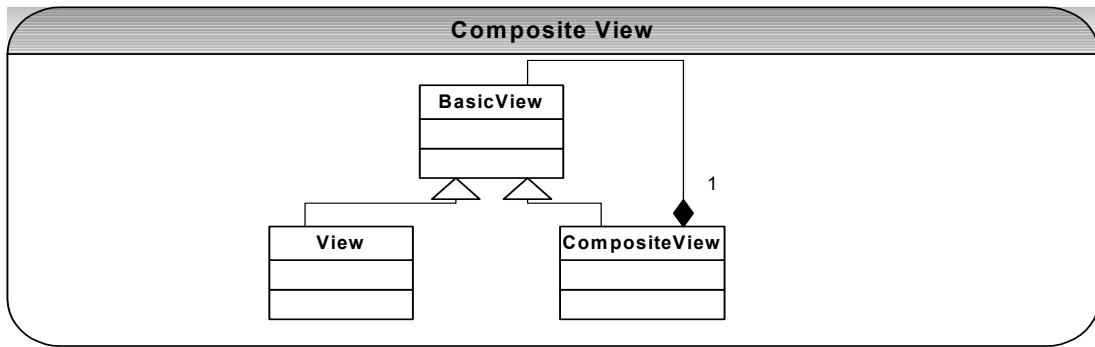
### **Benefits**

- Avoid duplicating the control logic like security check, flow control etc.
- Apply the common logic, which is shared by multiple requests in the Front controller.
- Separate the system processing logic from the view processing logic.
- Provides a controlled and centralized access point for your system.

**Q 25:** Briefly discuss the following patterns Composite view, View helper, Dispatcher view and Service to worker? Or explain J2EE design patterns? **DP**

**A 25:**

- **Composite View:** Creates an aggregate view from atomic sub-views. The Composite View entirely focuses on the View. The View is typically a JSP page, which has the HTML, JSP Tags etc. The JSP display pages mostly have a side bar, header, footer and main content area. These are the sub-views of the view. The sub-views can be either static or dynamic. The best practice is to have these sub-views as separate JSP pages and include them in the whole view. This will enable **reuse of JSP sub-views and improves maintainability** by having to change them at one place only.



- **View Helper:** When processing logic is embedded inside the controller or view it causes code duplication in all the pages. This causes maintenance problems, as any change to piece of logic has to be done in all the views. In the view helper pattern the view delegates its processing responsibilities to its helper classes. The helper classes **JavaBeans**: used to compute and store the presentation data and **Custom Tags**: used for computation of logic and displaying them iteratively complement each other.

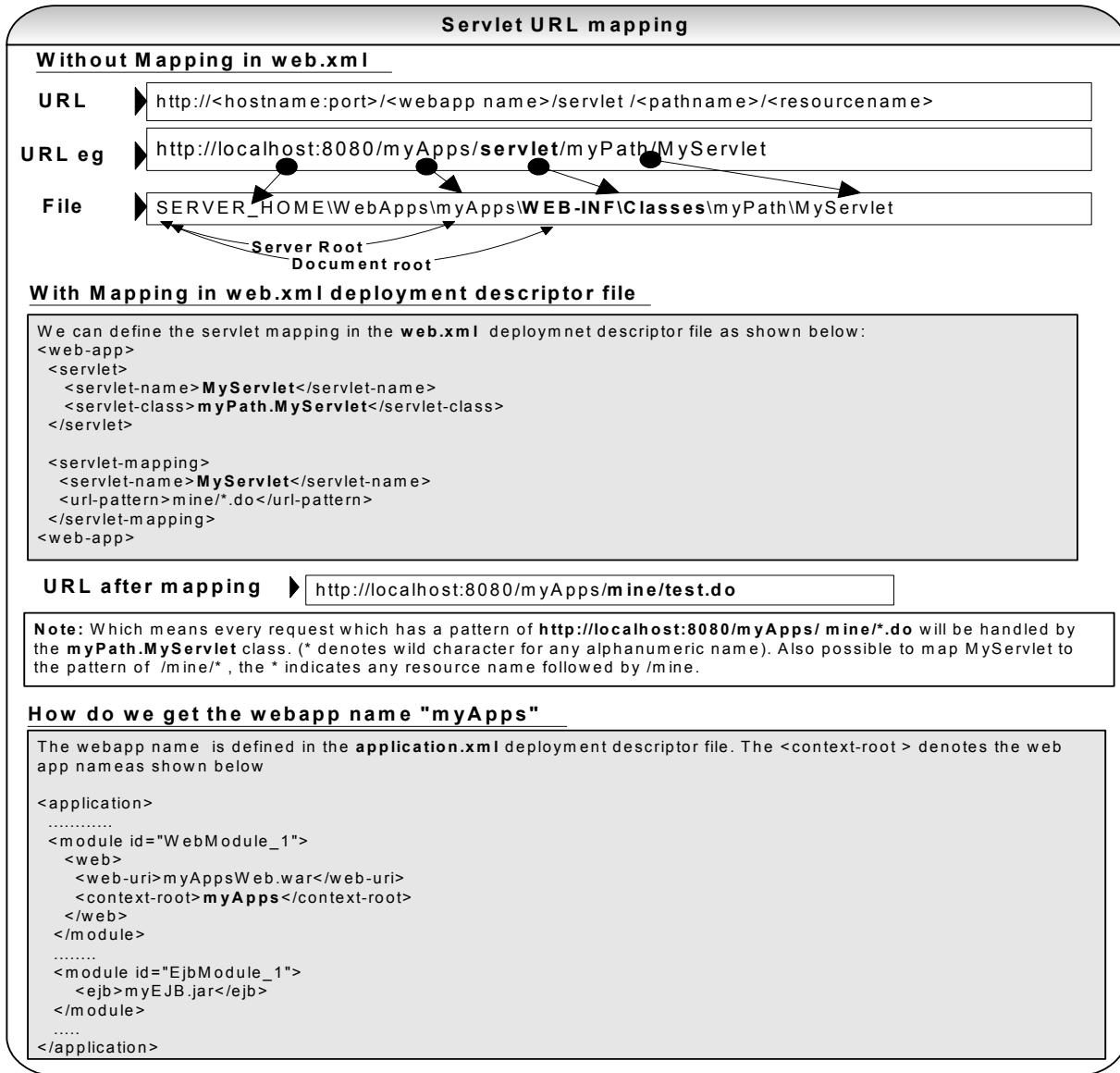
- Benefits** Avoids embedding programming logic in the views and facilitates division of labour between Java developers and Web page designers.
- **Service to Worker and Dispatcher View:** These two patterns are a combination of Front Controller and View Helper patterns with a *dispatcher* component. One of the responsibilities of a Front Controller is choosing a view and dispatching the request to an appropriate view. This behaviour can be partitioned into a separate component known as a *dispatcher*. But these two patterns differ in the way they suggest different division of responsibility among the components.

Service to Worker	Dispatcher View
<b>Service to Worker</b> Combines the front controller (Refer Q24 in Enterprise section) and dispatcher, with views and view helpers (refer Q25 in Enterprise section) to handle client requests and dynamically prepares the response. <ul style="list-style-type: none"> <li>▪ Controllers delegate the content retrieval to the view helpers, which populates the intermediate model content for the view.</li> <li>▪ Dispatcher is responsible for the view management and view navigation.</li> </ul>	<b>Dispatcher View</b> This pattern is structurally similar to the service to worker but the emphasis is on a different usage pattern. This combines the Front controller and the dispatcher with the view helpers but <ul style="list-style-type: none"> <li>▪ Controller <b>does not</b> delegate content retrieval to view helpers because this activity is deferred to view processing.</li> <li>▪ Dispatcher is responsible for the view management and view navigation.</li> </ul>

Promotes more up-front work by the front controller and dispatcher for the authentication, authorization, content retrieval, validation, view management and navigation.	Relatively has a lightweight front controller and dispatcher with minimum functionality and most of the work is done by the view.
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------

**Q 26:** Explain Servlet URL mapping? **SF**

**Q 26:**



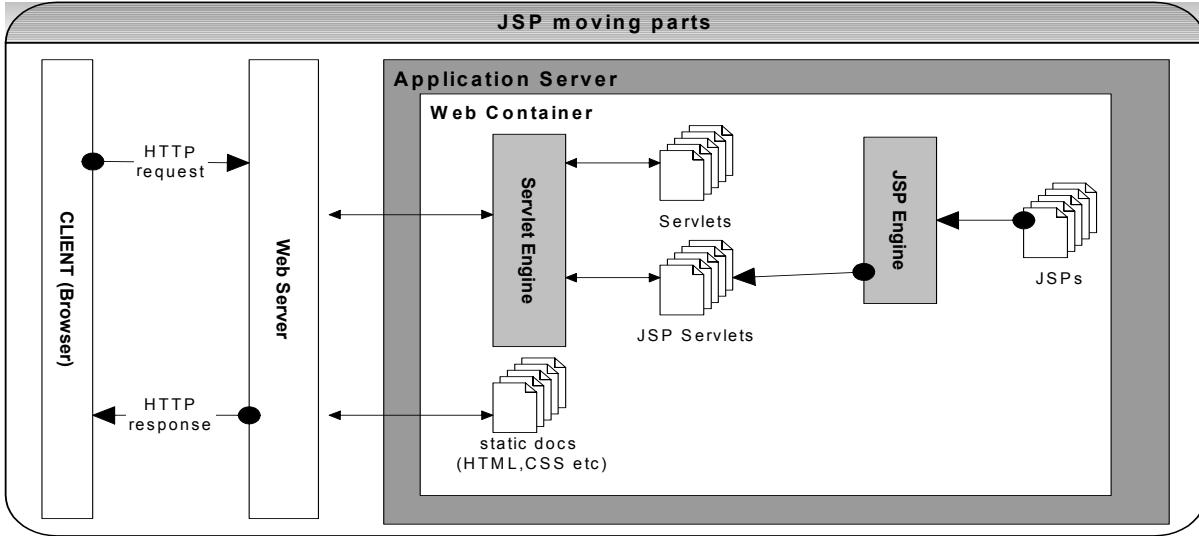
## Enterprise - JSP

**Q 27:** What is a JSP? What is it used for? What do you understand by the term JSP translation phase or compilation phase? **SF**

**A 27:** JSP (Java ServerPages) is an extension of the Java Servlet technology. JSP is commonly used as the **presentation** layer for combining HTML and Java code. While Java Servlet technology is capable of generating HTML with `out.println("<html>..... </html>")` statements, where `out` is a *PrintWriter*. This **process of embedding HTML code with escape characters is cumbersome and hard to maintain**. The JSP technology solves this by providing a level of abstraction so that the developer can use custom tags and action elements, which can speed up Web development and are easier to maintain.

As shown in the figure the JSPs have a **translation** or a **compilation** process where the JSP engine translates and compiles a JSP file into a JSP Servlet. The translated and compiled JSP Servlet moves to the **execution phase (run time)** where they can handle requests and send response.

Unless explicitly compiled ahead of time, JSP files are compiled the first time they are accessed. On large production sites, or in situations involving complicated JSP files, compilation may cause unacceptable delays to users first accessing the JSP page. The JSPs can be compiled ahead of time (ie **precompiled**) using application server tools/settings or by writing your own script.



**Q 28:** Explain the life cycle methods of a JSP? **SF**

**A 28:**

- **Pre-translated:** Before the JSP file has been translated and compiled into the Servlet.
- **Translated:** The JSP file has been translated and compiled as a Servlet.
- **Initialized:** Prior to handling the requests in the service method the container calls the `jsplInit()` to initialize the Servlet. Called only once per Servlet instance.
- **Servicing:** Services the client requests. Container calls this method for each request.
- **Out of service:** The Servlet instance is out of service. The container calls the `jsplDestroy()` method.

**Q 29:** What are the main elements of JSP? What are scriptlets? What are expressions? **SF**

**A 29:** There are two types of data in a JSP page.

- **Static part** (ie HTML, CSS etc), which gets copied directly to the response by the JSP Engine.
- **Dynamic part**, which contains anything that can be translated and compiled by the JSP Engine.

There are three types of dynamic elements. (**TIP:** remember **SAD** as an abbreviation for **S**cripting, **A**ction and **D**irective elements).

**Scripting Elements:** A JSP element that provides embedded Java statements. There are three types of scripting elements.

- **Declaration Element:** is the embedded Java declaration statement, which gets inserted at the Servlet class level.

```
<%! Calendar c = Calendar.getInstance(); %>
```

**Important:** declaring variables via this element is not thread-safe, because this variable ends up in the generated Servlet as an instance variable, not within the body of the `_jspservice()` method. Ensure their access is either read-only or synchronized.

- **Expression Element:** is the embedded Java expression, which gets evaluated by the service method.

```
<%= new Date()%>
```

- Scriptlet Elements:** are the embedded Java statements, which get executed as part of the service method.  
**(Note:** Not recommended to use Scriptlet elements because they don't provide reusability and maintainability. Use custom tags (like JSTL, JSF tags, etc) or beans instead).

```
<%
//Java codes
String userName=null;
userName=request.getParameter("userName");
%>
```

**Action Elements:** A JSP element that provides information for execution phase.

```
<jsp:useBean id="object_name" class="class_name"/>
<jsp:include page="scripts/login.jsp" />
```

**Directive Elements:** A JSP element that provides global information for the **translation** phase.

```
<%@ page import="java.util.Date" %>
<%@ include file="myJSP" %>
<%@ taglib uri="tagliburi" prefix="myTag" %>
```

**Q 30:** What are the different scope values or what are the different scope values for <jsp:usebean> ? **SF**

**A 30:**

Scope	Object	Comment
Page	PageContext	Available to the handling JSP page only.
Request	Request	Available to the handling JSP page or Servlet and forwarded JSP page or Servlet.
Session	Session	Available to any JSP Page or Servlet within the same session.
Application	Application	Available to all the JSP pages and Servlets within the same Web Application.

**Q 31:** What are the differences between static and a dynamic include? **SF DC**

**A 31:**

Static include <%@ include %>	Dynamic include <jsp:include .....>
During the translation or compilation phase all the included JSP pages are compiled into a single Servlet.	The dynamically included JSP is compiled into a separate Servlet. It is a separate resource, which gets to process the request, and the content generated by this resource is included in the JSP response.
No run time performance overhead.	Has run time performance overhead.

**Which one to use:** Use "static includes" when a JSP page does not change very often. For the pages, which change frequently, use dynamic includes. JVM has a 64kb limit on the size of the method and the entire JSP page is rendered as a single method. **If a JSP page is greater than 64kb, this probably indicates poor implementation.** When this method reaches its limit of 64kb it throws an error. This error can be overcome by splitting the JSP files and including them dynamically (i.e. using <jsp:include.....>) because the dynamic includes generate separate JSP Servlet for each included file.

**Note:** The "dynamic include" (jsp:include) has a **flush** attribute. This attribute indicates whether the buffer should be flushed before including the new content. In JSP 1.1 you will get an error if you omit this attribute. In JSP 1.2 you can omit this attribute because the flush attribute defaults to false.

**Q 32:** What are implicit objects and list them? **SF**

**A 32:** Implicit objects are the objects that are available for the use in JSP documents without being declared first. These objects are parsed by the JSP engine and inserted into the generated Servlet. The implicit objects are:

Implicit object	Scope	comment
request	Request	request
response	Page	response
pageContext	Page	page environment
session	Session	session
application	Application	same as ServletContext
out	Page	writing to the outputstream
config	Page	same as ServletConfig
page	Page	this page's Servlet
exception	Page	exception created on this page.

**Note:** Care should be taken not to name your objects the same name as the implicit objects. If you have your own object with the same name, then the implicit objects take precedence over your own object.

**Q 33:** Explain hidden and output comments? **SF**

**A 33:** An output comment is a comment that is sent to the client where it is viewable in the browser's source. **CO**

```
<!--This is a comment which is sent to the client-->
```

A hidden comment documents a JSP page but does not get sent to the client. The JSP engine ignores a hidden comment, and does not process any code within hidden comment tags.

```
<%-- This comment will not be visible to the client --%>
```

**Q 34:** Is JSP variable declaration thread safe? **CI**

**A 34:** No. The declaration of variables in JSP is not thread-safe, because the declared variables end up in the generated Servlet as an instance variable, not within the body of the `_jspService()` method.

**The following declaration is not thread safe:** because these are declarations, and will only be evaluated once when the page is loaded

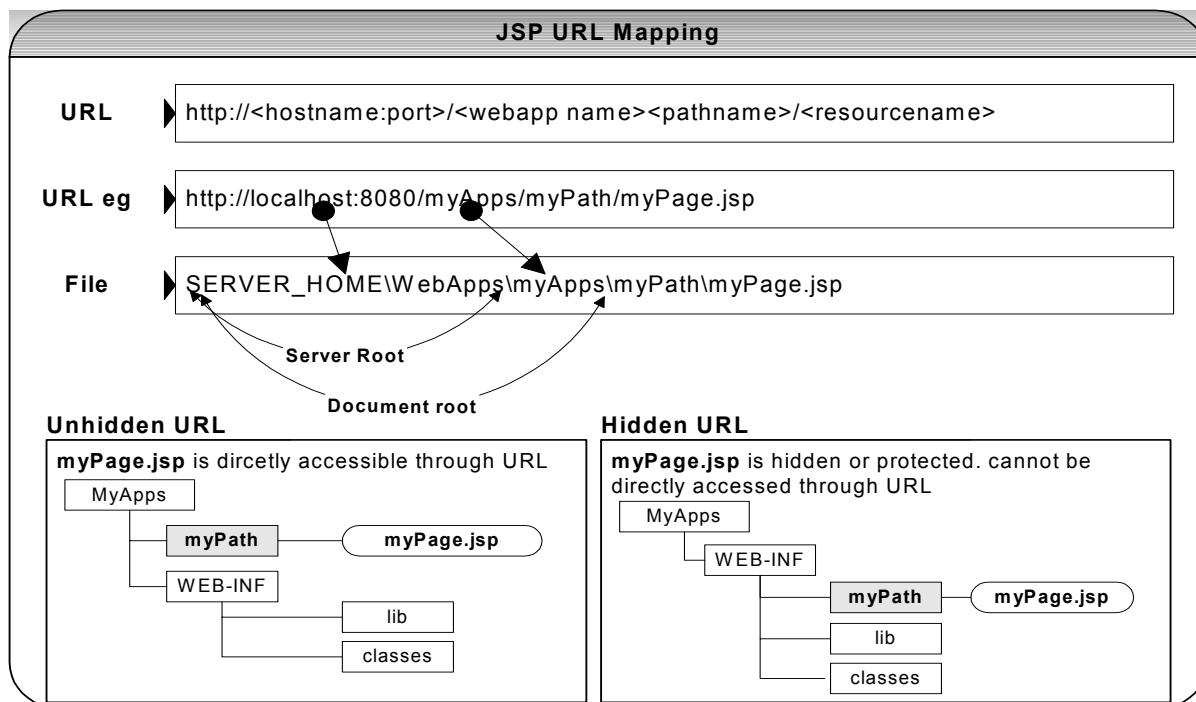
```
<%! int a = 5 %>
```

**The following declaration is thread safe:** because the variables declared inside the scriptlets have the local scope and not shared.

```
<% int a = 5 %>
```

**Q 35:** Explain JSP URL mapping? What is URL hiding or protecting the JSP page? **SF SE**

**A 35:** As shown in the figure, the JSP resources usually reside directly or under subdirectories (e.g. `myPath`) of the **document root**, which are **directly accessible** to the user through the URL. If you want to protect your Web resources then hiding the JSP files behind the `WEB-INF` directory can protect the JSP files, css (cascading style sheets) files, Java Script files, pdf files, image files, html files etc from direct access. The request should be made to a servlet who is responsible for authenticating and authorising the user before returning the protected JSP page or its resources.

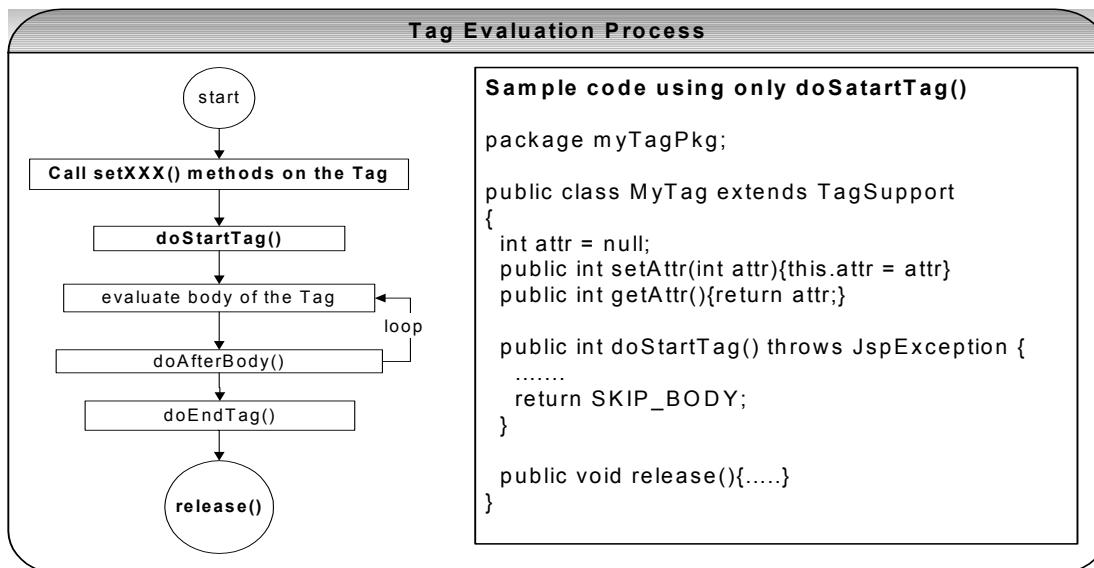


**Q 36:** What are custom tags? Explain how to build custom tags? **SF**

**A 36:** Custom JSP tag is a tag you define. You define how a tag, its attributes and its body are interpreted, and then group your tags into collections called tag libraries that can be used in any number of JSP files. So basically it is a reusable and extensible JSP only solution. The pre-built tags also can speed up Web development. **CO**

#### STEP: 1

Construct the Tag handler class that defines the behaviour.



#### STEP: 2

The Tag library descriptor file (\*.tld) maps the XML element names to the tag implementations. The code sample **MyTagDesc.tld** is shown below:

```

<taglib>
<tag>
 <name>tag1</name>
 <tagclass>myTagPkg.MyTag</tagclass>
 <bodycontent>empty</bodycontent>
 <attribute>
 <name>attr</name>
 <required>false</required>
 <rtpxvalue>false</rtpxvalue>
 </attribute>
</tag>
</taglib>

```

#### STEP: 3

The web.xml deployment descriptor maps the URI to the location of the \*.tld (Tag Library Descriptor) file. The code sample web.xml file is shown below:

```

<web-app>
<taglib>
 <taglib-uri>/WEB-INF/MyTagURI</taglib-uri>
 <taglib-location>/WEB-INF/tags/MyTagDesc.tld</taglib-location>
</taglib>
</web-app>

```

#### STEP: 4

The JSP file declares and then uses the tag library as shown below:

```

<%@ taglib uri="/WEB-INF/ MyTagURI" prefix="myTag" %>
< myTag:tag1 attr="abc"></ myTag:tag1> or < myTag:tag1 attr="abc" />

```

**Q 37:** What is a TagExtraInfo class? **SF**

**A 37:** A TagExtraInfo class provides extra information about tag attributes to the JSP container at translation time.

- **Returns information about the scripting variables** that the tag makes available to the rest of the JSP page to use. The method used is:

```
VariableInfo[] getVariableInfo(TagData td)
```

**Example**

```
<html>
 <myTag:addObjectsToArray name="myArray" />
 <myTag:displayArray name="myArray" />
</html>
```

Without the use of TagExtraInfo, if you want to manipulate the attribute *myArray* in the above code in a scriptlet it will not be possible. This is because it does not place the *myArray* object on the page. You can still use `pageContext.getAttribute()` but that may not be a cleaner approach because it relies on the page designer to correctly cast to object type. The *TagExtraInfo* can be used to make items stored in the `pageContext` via `setAttribute()` method available to the scriptlet as shown below.

```
<html>
 <myTag:addObjectsToArray name="myArray" />
 <%-- scriptlet code %>
 <% for(int i=0; i<myArray.length;i++){
 html += + myArray[i] + ;
 %>
</html>
```

- **Validates the attributes passed to the Tag at translation time.**

**Example** It can validate the *myArray* array list to have not more than 100 objects. The method used is:

```
boolean isValid(TagData data)
```

**Q 38:** What is the difference between custom JSP tags and JavaBeans? **SF**

**A 38:** In the context of a JSP page, both accomplish similar goals but the differences are:

Custom Tags	JavaBeans
Can manipulate JSP content.	Can't manipulate JSP content.
Custom tags can simplify the complex operations much better than the bean can. But require a bit more work to set up.	Easier to set up.
Used only in JSPs in a relatively self-contained manner.	Can be used in both Servlets and JSPs. You can define a bean in one Servlet and use them in another Servlet or a JSP page.

**JavaBeans declaration and usage example:** **CO**

```
<jsp:useBean id="identifier" class="packageName.className"/>
<jsp:setProperty name="identifier" property="classField" value="someValue" />
<jsp:getProperty name="identifier" property="classField" /> <%=identifier.getClassField() %>
```

**Q 39:** Tell me about JSP best practices? **BP**

**A 39:**

- **Separate HTML code from the Java code:** Combining HTML and Java code in the same source code can make the code less readable. Mixing HTML and scriptlet will make the code extremely difficult to read and maintain. The display or behaviour logic can be implemented as a custom tags by the Java developers and Web designers can use these Tags as the ordinary XHTML tags.
- **Place data access logic in JavaBeans:** The code within the JavaBean is readily accessible to other JSPs and Servlets.

- **Factor shared behaviour out of Custom Tags into common JavaBeans classes:** The custom tags are not used outside JSPs. To avoid duplication of behaviour or business logic, move the logic into JavaBeans and get the custom tags to utilize the beans.
- **Choose the right “include” mechanism:** What are the differences between static and a dynamic include? Using includes will improve code reuse and maintenance through modular design. Which one to use? Refer **Q31** in Enterprise section.
- **Use style sheets** (e.g. css), **template mechanism** (e.g. struts tiles etc) and **appropriate comments** (both hidden and output comments).

**Q 40:** How will you avoid scriptlet code in JSP? **BP**

**A 40:** Use JavaBeans or Custom Tags instead.

### Enterprise - JDBC

**Q 41:** What is JDBC? How do you connect to a database? **SF**

**A 41:** JDBC stands for **Java Database Connectivity**. It is an API which provides easy connection to a wide range of databases. To connect to a database we need to load the appropriate driver and then request for a connection object. The Class.forName("....") will load the driver and register it with the DriverManager (Refer **Q4** in Java section for dynamic class loading).

```
Class.forName("oracle.jdbc.driver.OracleDriver");
String url = jdbc:oracle:thin:@hostname:1526:myDB;
Connection myConnection = DriverManager.getConnection(url, "username", "password");
```

The **DataSource** interface provides an alternative to the **DriverManager** for making a connection. **DataSource** makes the code more portable than **DriverManager** because it works with JNDI and it is created, deployed and managed separately from the application that uses it. If the **DataSource** location changes, then there is no need to change the code but change the configuration properties in the server. This makes your application code easier to maintain. **DataSource** allows the use of connection pooling and support for distributed transactions. A **DataSource** is not only a database but also can be a file or a spreadsheet. A **DataSource** object can be bound to JNDI and an application can retrieve and use it to make a connection to the database. J2EE application servers provide tools to define your **DataSource** with a JNDI name. When the server starts it loads all the **DataSources** into the Application Server's JNDI service.

DataSource configuration properties are shown below:

- **JNDI Name** → jdbc/myDataSource
- **URL** → jdbc:oracle:thin:@hostname:1526:myDB
- **UserName, Password**
- **Implementation classname** → oracle.jdbc.pool.OracleConnectionPoolDataSource
- **Classpath** → ora\_jdbc.jar
- **Connection pooling** settings like → minimum pool size, maximum pool size, connection timeout, statement cache size etc.

Once the **DataSource** has been set up, then you can get the connection object as follows:

```
Context ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("jdbc/myDataSource");
Connection myConnection = ds.getConnection("username","password");
```

In a basic implementation a **Connection** obtained from a **DataSource** and a **DriverManager** are identical. But, **DataSource is recommended because of its better portability**.

**Design Pattern:** JDBC architecture decouples an abstraction from its implementation so that the implementation can vary independent of the abstraction. This is an example of the **bridge design pattern**. The JDBC API provides the abstraction and the JDBC drivers provide the implementation. New drivers can be plugged-in to the JDBC API without changing the client code.

**Q 42:** What are JDBC Statements? What are different types of statements? How can you create them? **SF**

**A 42:** A **statement** object is responsible for sending the SQL statements to the Database. Statement objects are created from the connection object and then executed. **CO**

```
Statement stmt = myConnection.createStatement();
ResultSet rs = stmt.executeQuery("SELECT id, name FROM myTable where id =1245");// to read
Or
stmt.executeUpdate("INSERT INTO (field1,field2) values (1,3)");// to insert/update/delete/create table
```

The types of statements are:

- **Statement** (regular statement as shown above)
- **PreparedStatement** (more efficient than statement due to pre-compilation of SQL)
- **CallableStatement** (to call stored procedures on the database)

To use prepared statement:

```
PreparedStatement prepStmt =
 myConnection.prepareStatement("SELECT id, name FROM myTable where id = ? ");
prepStmt.setInt(1, 1245);
```

Callable statements are used for calling stored procedures.

```
CallableStatement calStmt = myConnection.prepareCall("{call PROC_SHOWMYBOOKS}");
ResultSet rs = cs.executeQuery();
```

**Q 43:** What is a **Transaction**? What does **setAutoCommit** do? **TI PI**

**A 43:** A transaction is a set of operations that should be completed as a unit. If one operation fails then all the other operations fail as well. For example if you transfer funds between two accounts there will be two operations in the set

1. Withdraw money from one account.
2. Deposit money into other account.

These two operations should be completed as a single unit. Otherwise your money will get lost if the withdrawal is successful and the deposit fails. There are four characteristics (**ACID** properties) for a Transaction.

Atomicity	Consistency	Isolation	Durability
All the individual operations should either complete or fail.	The design of the transaction should update the database correctly.	Prevents data being corrupted by concurrent access by two different sources. It keeps transactions isolated or separated from each other until they are finished.	Ensures that the database is definitely updated once the Transaction is completed.

Transactions maintain data integrity. A transaction has a beginning and an end like everything else in life. The **setAutocommit(...)**, **commit()** and **rollback()** are used for marking the transactions (known as transaction demarcation). When a connection is created, it is in **auto-commit** mode. This means that each individual SQL statement is treated as a transaction and will be automatically committed immediately after it is executed. The way to allow two or more statements to be grouped into a transaction is to **disable** auto-commit mode: **CO**

```
try{
 Connection myConnection = dataSource.getConnection();

 // set autoCommit to false
 myConnection .setAutoCommit(false);

 withdrawMoneyFromFirstAccount(.); //operation 1
 depositMoneyIntoSecondAccount(.); //operation 2

 myConnection .commit();
}
catch(Exception sqle){
 try{
 myConnection .rollback();
 }catch(Exception e){}
}
finally{
 try{if(conn != null) {conn.close();}} catch(Exception e) {}
}
```

The above code ensures that both operation 1 and operation 2 succeed or fail as an atomic unit and consequently leaves the database in a consistent state. Also turning auto-commit off will provide better performance.

**Q 44:** What is the difference between JDBC-1.0 and JDBC-2.0? What are Scrollable ResultSets, Updateable ResultSets, RowSets, and Batch updates? **SF**

**A 44:** JDBC2.0 has the following additional features or functionality:

JDBC 1.0	JDBC 2.0
With JDBC-1.0 the ResultSet functionality was limited. There was no support for updates of any kind and scrolling through the ResultSets was forward only (no going back)	With JDBC 2.0 ResultSets are updateable and also you can move forward and backward.  <b>Example</b> This example creates an updateable and scroll-sensitive ResultSet  Statement stmt = myConnection.createStatement(resultSet.TYPE_SCROLL_SENSITIVE, resultSet.CONCUR_UPDATEABLE)
With JDBC-1.0 the statement objects submits updates to the database individually within same or separate transactions. This is very inefficient large amounts of data need to be updated.	With JDBC-2.0 statement objects can be grouped into a batch and executed at once. We call addBatch() multiple times to create our batch and then we call executeBatch() to send the SQL statements off to database to be executed as a batch (this minimises the network overhead).  <b>Example</b>  Statement stmt = myConnection.createStatement(); stmt.addBatch("INSERT INTO myTable1 VALUES (1,'ABC')"); stmt.addBatch("INSERT INTO myTable1 VALUES (2,'DEF')"); stmt.addBatch("INSERT INTO myTable1 VALUES (3,'XYZ')"); ... int[] countInserts = stmt.executeBatch();
-	The JDBC-2.0 optional package provides a RowSet interface, which extends the ResultSet. One of the implementations of the RowSet is the CachedRowSet, which can be considered as a disconnected ResultSet.

**Q 45:** How to avoid the “running out of cursors” problem? **DC PI MI**

**A 45:** A database can run out of cursors if the connection is not closed properly or the DBA has not allocated enough cursors. In a Java code it is essential that we close all the valuable resources in a try{} and finally{} block. The finally{} block is always executed even if there is an exception thrown from the catch {} block. So the resources like connections and statements should be closed in a finally {} block. **CO**

**Try{} Finally {} blocks to close Exceptions**

<p><b>Wrong Approach -</b></p> <p>Connections and statements will not be closed if there is an exception:</p> <pre>public void executeSQL() throws SQLException{     Connection con = DriverManager.getConnection(.....);     ....     Statement stmt = con.createStatement();     ....     //line 20 where exception is thrown     ResultSet rs = stmt.executeQuery("SELECT * from myTable");     ...     rs.close();     stmt.close();     con.close(); }</pre> <p><b>Note:</b> if an exception is thrown at line 20 then the close() statements are never reached.</p>	<p><b>Right Approach -</b></p> <pre>public void executeSQL() throws SQLException{     try{         Connection con = DriverManager.getConnection(.....);         ....         Statement stmt = con.createStatement();         ....         //line 20 where exception is thrown         ResultSet rs = stmt.executeQuery("SELECT * from myTable");         ...     }     finally{         try{             if(rs != null) rs.close();             if(stmt != null) stmt.close();             if(con != null) con.close();         }         catch(Exception e){}     } }</pre> <p><b>Note:</b> if an exception is thrown at line 20 then the finally clause is called before the exception is thrown to the method.</p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Q 46:** What is the difference between statements and prepared statements? **SF PI SE BP**

**A 46:**

- Prepared statements offer better performance, as they are **pre-compiled**. Prepared statements reuse the same **execution plan** for different arguments rather than creating a new execution plan every time. Prepared statements use bind arguments, which are sent to the database engine. This allows mapping different requests with same prepared statement but different arguments to execute the same execution plan.
- Prepared statements are more secure because they use bind variables, which can prevent SQL injection attack.

The most common type of SQL injection attack is SQL manipulation. The attacker attempts to modify the SQL statement by adding elements to the WHERE clause or extending the SQL with the set operators like UNION, INTERSECT etc.

**Example** Let us look at the following SQL:

```
SELECT * FROM users where username='bob' AND password='xyfdsw' ;
```

The attacker can manipulate the SQL as follows

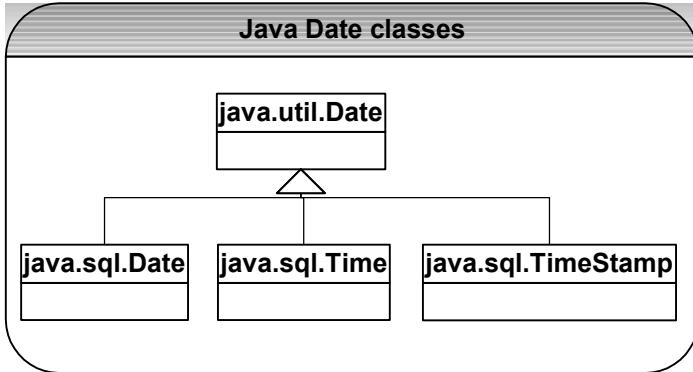
```
SELECT * FROM users where username='bob' AND password='xyfdsw' OR 'a' = 'a' ;
```

The above "WHERE" clause is always true because of the operator precedence. The PreparedStatement can prevent this by using bind variables:

```
String strSQL = SELECT * FROM users where username=? AND password=?;
PreparedStatement pstmt = myConnection.prepareStatement(strSQL);
pstmt.setString(1,"bob");
pstmt.setString(2, "xyfdsw");
pstmt.execute();
```

**Q 47:** Explain differences among java.util.Date, java.sql.Date, java.sql.Time, and java.sql.Timestamp? **SF**

**A 47:** As shown below all the sql Date classes extend the util Date class.



**java.util.Date** - class supports both the Date (ie year/month/date etc) and the Time (hour, minute, second, and millisecond) components.

**java.sql.Date** - class supports only the Date (ie year/month/date etc) component. The hours, minutes, seconds and milliseconds of the Time component will be set to zero in the particular time zone with which the instance is associated.

**java.sql.Time** - class supports only Time (ie hour, minute, second, and millisecond) component. The date components should be set to the "zero epoch" value of January 1, 1970 and should not be accessed.

**java.sql.TimeStamp** – class supports both Date (ie year/month/date etc) and the Time (hour, minute, second, millisecond and **nanosecond**) components.

**Note:** the subtle difference between **java.util.Date** and **java.sql.Date**.

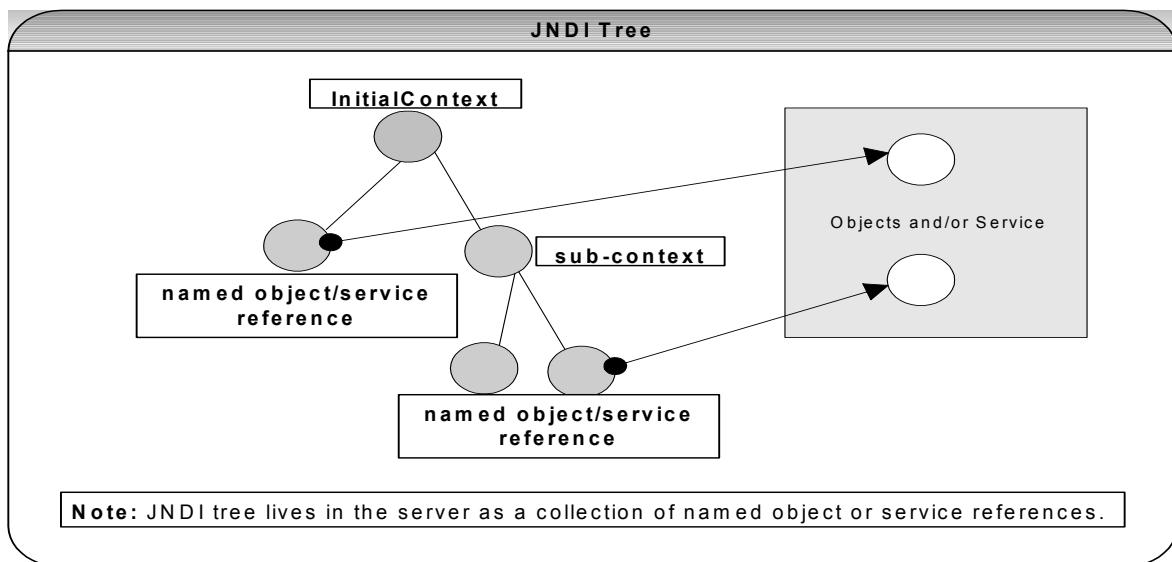
To keep track of time Java counts the number of milliseconds from January 1, 1970 and stores it as a long value in `java.util.Date` class. The `GregorianCalendar` class provides us a way to represent an arbitrary date. The `GregorianCalendar` class also provides methods for manipulating dates.

### Enterprise – JNDI & LDAP

**Q 48:** What is JNDI? And what are the typical uses within a J2EE application? **SF**

**A 48:** JNDI stands for Java Naming and Directory Interface. It provides a generic interface to LDAP (Lightweight Directory Access Protocol) and other directory services like NDS, DNS (Domain Name System) etc. It provides a means for an application to locate components that exist in a name space according to certain attributes. A J2EE application component uses JNDI interfaces to look up and reference system-provided and user-defined objects in a component environment. JNDI is not specific to a particular naming or directory service. It can be used to access many different kinds of systems including file systems.

The JNDI API enables applications to look up objects such as DataSources, EJBs, MailSessions and JMS by name. The Objects can be loaded into the JNDI tree using a J2EE application server's administration console. To load an object in a JNDI tree, choose a name under which you want the object to appear in a JNDI tree. J2EE deployment descriptors indicate the placement of J2EE components in a JNDI tree.



The parameters you have to define for JNDI service are as follows:

- The name service provider class name (WsnInitialContext for Websphere).

```
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY, "com.ibm.websphere.naming.WsnInitialContextFactory");
```

- The provider URL :

- The name service hostname.
- The name service port number.

```
env.put(Context.PROVIDER_URL, "iiop://localhost:1050");
Context ctx = new InitialContext(env);
```

JNDI is like a file system or a Database.

File System	JNDI	Database
File system starts with a <b>mounted drive</b> like c:\	JNDI starts with an <b>InitialContext</b> . i.e. new <code>InitialContext()</code> .	<b>Database instance</b>

Uses a <b>subdirectory</b> . C:\subdir1	Navigate to a <b>sub-context</b> . e.g. Subcontext1	<b>Tablespace</b>
Access a <b>subdirectory</b> c:\subdir1\subdir2	Drill down through other <b>sub-contexts</b> . e.g. subcontext1/subcontext2	<b>Table</b>
Access a <b>file</b> . C:\subdir1\subdir2\myFile	Access an <b>object or a service</b> . New InitialContext().lookup("objectName");	<b>Data</b>
<b>Example:</b>  c:\subdir1\subdir2\myFile	<b>Example:</b>  iiop://myserver:2578/subcontext1.subcontext2.o bjectName	<b>Example:</b>  Select * from demo.myTable

**Q 49:** Explain the difference between the look up of “java:comp/env/ejb/MyBean” and “ejb/MyBean”? **SF**

**A 49:**

<b>java:comp/env/ejb/MyBean</b>	<b>ejb/MyBean</b>
This is a logical reference, which will be used in your code.	This is a physical reference where an object will be mapped to in a JNDI tree.

The logical reference (or alias) **java:comp/env/ejb/MyBean** is the recommended approach because you cannot guarantee that the physical JNDI location (**ejb/MyBean**) you specify in your code will be available. Your code will break if the physical location is changed. The deployer will not be able to modify your code. Logical references solve this problem by binding the logical name to the physical name in the application server. The logical names will be declared in the deployment descriptors (web.xml and/or ejb-jar.xml) as follows and these will be mapped to physical JNDI locations in the application server specific deployment descriptors.

To look up a JDBC resource from either WEB (web.xml) or EJB (ejb-jar.xml) tier, the deployment descriptor should have the following entry:

```
<resource-ref>
 <description>The DataSource</description>
 <res-ref-name>jdbc/MyDataSource</res-ref-name>
 <res-type>javax.sql.DataSource</res-type>
 <res-auth>Container</res-auth>
</resource-ref>
```

This will make full logical path to the bean  
as:  
**java:comp/env/jdbc/MyDataSource**

To use it:

```
Context ctx = new InitialContext();
Object ref = ctx.lookup(java:comp/env/jdbc/MyDataSource);
```

To look up EJBs from another EJB or a WEB module, the deployment descriptor should have the following entry:

```
<ejb-ref>
 <description>myBean</description>
 <ejb-ref-name>ejb/MyBean</ejb-ref-name>
 <ejb-ref-type>Entity</ejb-ref-type>
 <ejb-link>Region</ejb-link>
 <home>com.MyBeanHome</home>
 <remote>com.MyBean</remote>
</ejb-ref>
```

This will make full logical path to the bean  
as:  
**java:comp/env/ejb/MyBean**

To use it:

```
Context ctx = new InitialContext();
Object ref = ctx.lookup(java:comp/env/ejb/MyBean);
```

**Q 50:** What is a JNDI InitialContext? **SF**

**A 50:** All naming operations are relative to a context. The *InitialContext* implements the *Context* interface and provides an **entry point** for the resolution of names.

**Q 51:** What is an LDAP server? And what is it used for in an enterprise environment? **SF SE**

**A 51:** LDAP stands for **L**ightweight **D**irectory **A**ccess **P**rotocol. This is an extensible open network protocol standard that provides access to distributed directory services. LDAP is an Internet standard for directory services that run on TCP/IP. Under OpenLDAP and related servers, there are two servers – **slapd**, the LDAP daemon where the queries are sent to and **slurpd**, the replication daemon where data from one server is pushed to one or more

slave servers. By having multiple servers hosting the same data, you can increase reliability, scalability, and availability.

- It defines the operations one may perform like search, add, delete, modify, change name
- It defines how operations and data are conveyed.

LDAP has the potential to consolidate all the existing application specific information like user, company phone and e-mail lists. This means that the change made on an LDAP server will take effect on every directory service based application that uses this piece of user information. The variety of information about a new user can be added through a single interface which will be made available to Unix account, NT account, e-mail server, Web Server, Job specific news groups etc. When the user leaves his account can be disabled to all the services in a single operation.

So LDAP is most useful to provide “white pages” (e.g. names, phone numbers, roles etc) and “yellow pages” (e.g. location of printers, application servers etc) like services. Typically in a J2EE application environment it will be used to authenticate and authorise users.

#### **Why use LDAP when you can do the same with relational database (RDBMS)?**

In general LDAP servers and RDBMS are designed to provide different types of services. LDAP is an open standard access mechanism, so an RDBMS can talk LDAP. However the servers, which are built on LDAP, are **optimized for read access** so likely to be much faster than RDBMS in providing read access. So in a nutshell, **LDAP is more useful when the information is often searched but rarely modified**. (Another difference is that RDBMS systems store information in rows of tables whereas LDAP uses object oriented hierarchies of entries.).

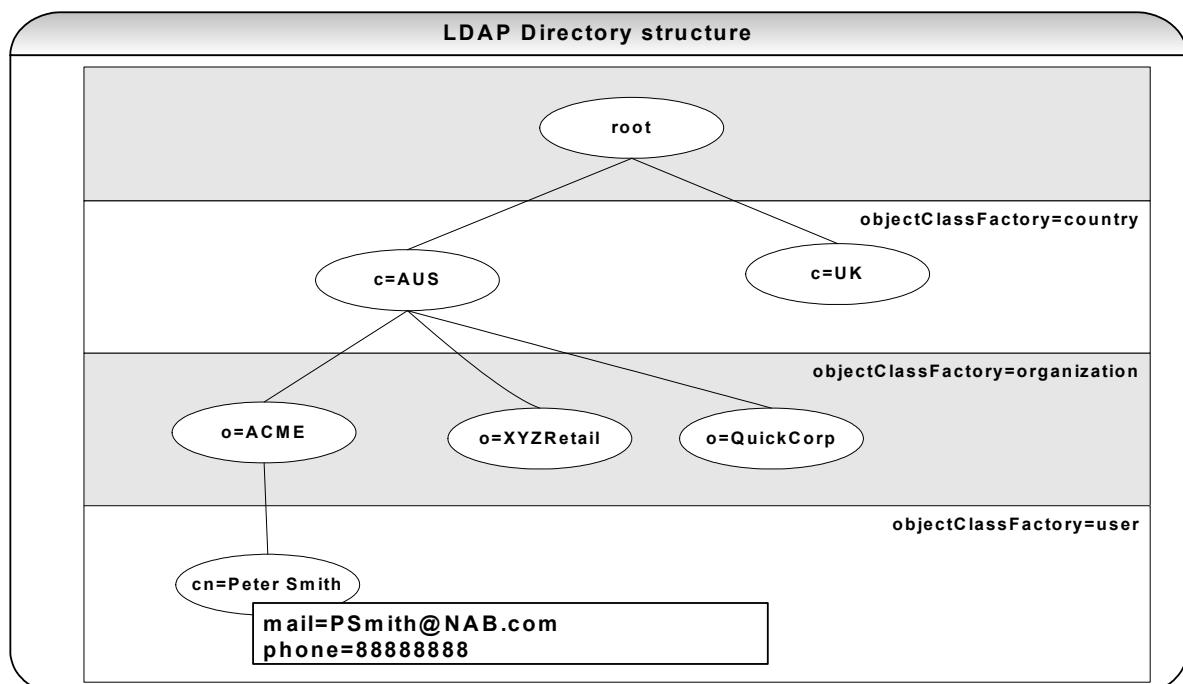
#### **Key LDAP Terms:**

**DIT:** Directory Information Tree. Hierarchical structure of entries, those make up a directory.

**DN:** Distinguished Name. This uniquely identifies an entry in the directory. A **DN is made up of relative DNs** of the entry and each of entry's parent entries up to the root of the tree. DN is read from right to left and commas separate these names. For example '**cn=Peter Smith, o=ACME, c=AUS**'.

**objectClass:** An *objectClass* is a formal definition of a specific kind of objects that can be stored in the directory. An ObjectClass is a distinct, named set of attributes that represent something concrete such as a user, a computer, or an application.

**LDAP URL:** This is a string that specifies the location of an LDAP resource. An LDAP URL consists of a server host and a port, search scope, **baseDN**, filter, attributes and extensions. Refer to diagram below:



So the complete distinguished name for bottom left entry (ie Peter Smith) is **cn=Peter Smith, o=ACME, c=AUS**. Each entry must have at least one attribute that is used to name the entry. To manage the part of the LDAP directory we should specify the highest level parent distinguished names in the server configuration. These distinguished names are called **suffixes**. The server can access all the objects that are below the specified suffix in the hierarchy. For example in the above diagram to answer queries about 'Peter Smith' the server should have the **suffix** of 'o=ACME, c=AUS'. So we can look for "Peter Smith" by using the following distinguished name:

```
cn=Peter Smith, o=ACME, c=AUS //where o=ACME, c=AUS is the suffix
```

**LDAP schema:** defines rules that specify the types of objects that a directory may contain and the required optional attributes that entries of different types should have.

**Filters:** In LDAP the basic way to retrieve data is done with filters. There is a wide variety of operators that can be used as follows: & (and), | (or), ! (not), ~= (approx equal), >= (greater than or equal), <= (less than or equal), \* (any) etc.

```
(& (uid=a*) (uid=*l))
```

**So where does JNDI fit into this LDAP?** JNDI provides a standard API for interacting with naming and directory services using a service provider interface (SPI), which is analogous to JDBC driver. To connect to an LDAP server, you must obtain a reference to an object that implements the **DirContext**. In most applications, this is done by using an *InitialDirContext* object that takes a Hashtable as an argument:

```
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
env.put(Context.PROVIDER_URL, "ldap://localhost:387");
env.put(Context.SECURITY_AUTHENTICATION, "simple");
env.put(Context.SECURITY_PRINCIPAL, "cn=Directory Manager");
env.put(Context.SECURITY_CREDENTIALS, "myPassword");
DirContext ctx = new InitialDirContext(env);
```

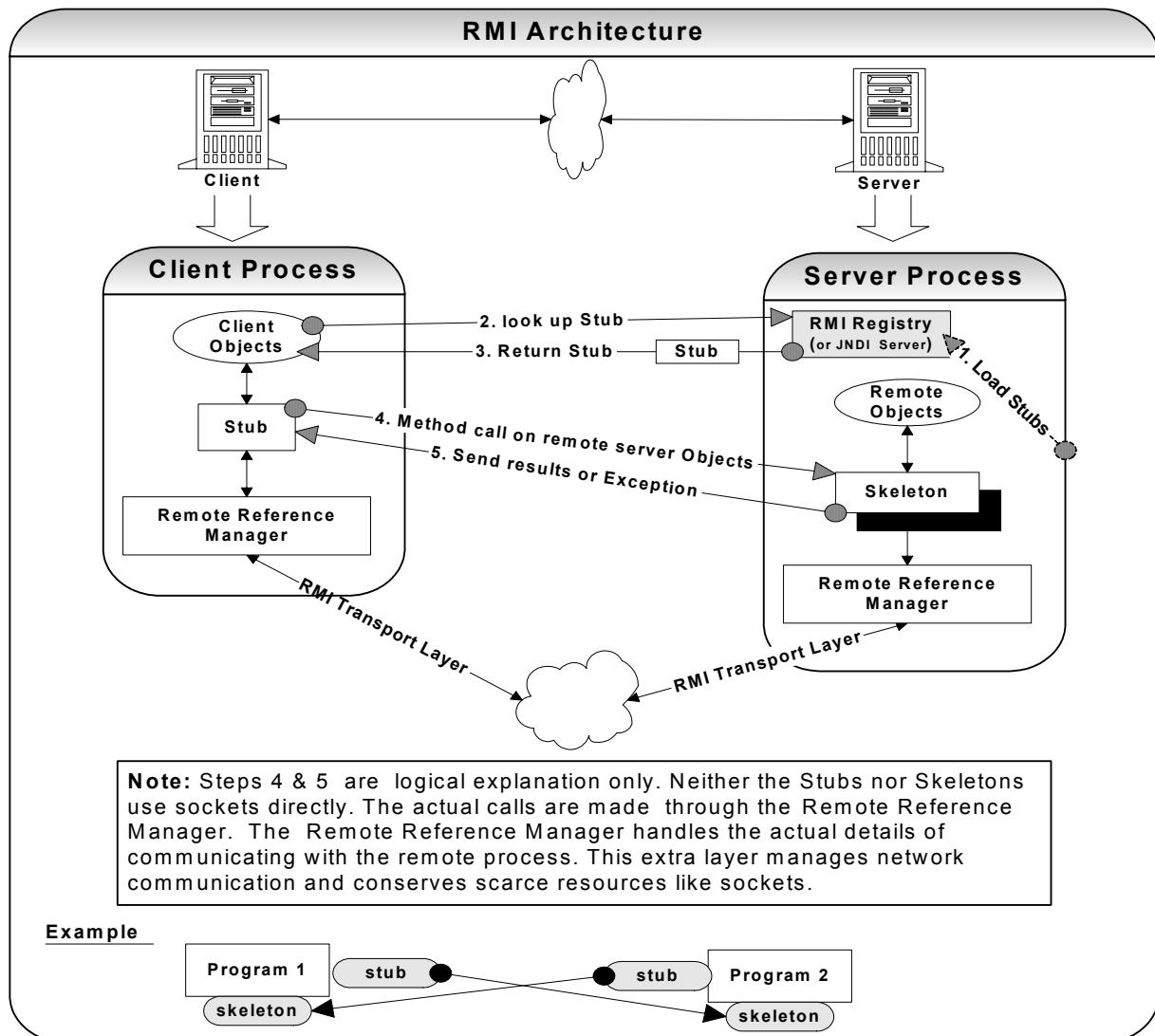
## Enterprise - RMI

**Q 52:** Explain the RMI architecture? **SF**

**A 52:** Java Remote Method Invocation (RMI) provides a way for a Java program on one machine to communicate with objects residing in different JVMs (or processes or address spaces). The important parts of the RMI architecture are the stub class, object serialization and the skeleton class. RMI uses a layered architecture where each of the layers can be enhanced without affecting the other layers. The layers can be summarised as follows:

- **Application Layer:** The client and server program
- **Stub & Skeleton Layer:** Intercepts method calls made by the client. Redirects these calls to a remote RMI service.
- **Remote Reference Layer:** Sets up connections to remote address spaces, manages connections, and understands how to interpret and manage references made from clients to the remote service objects.
- **Transport layer:** Based on TCP/IP connections between machines in a network. It provides basic connectivity, as well as some firewall penetration strategies.

**Design pattern:** RMI stub classes provide a reference to a skeleton object located in a different address space on the same or different machine. This is a typical example of a **proxy design pattern** (i.e. remote proxy), which makes an object executing in another JVM appear like a local object. In JDK 5.0 and later, the RMI facility uses **dynamic proxies** instead of generated stubs, which makes RMI easier to use. Refer **Q11** in "How would you about..." section for a more detailed discussion on proxy design pattern and dynamic proxies.



RMI runtime steps (as shown in the diagram above) involved are:

- Step 1:** Start RMI registry and then the RMI server. Bind the remote objects to the RMI registry.
- Step 2:** The client process will look up the remote object from the RMI registry.
- Step 3:** The lookup will return the stub to the client process from the server process.
- Step 4:** The client process will invoke method calls on the stub. The stub calls the skeleton on the server process through the RMI reference manager.
- Step 5:** The skeleton will execute the actual method call on the remote object and return the result or an exception to the client process via the RMI reference manager and the stub.

**Q 53:** What is a remote object? Why should we extend **UnicastRemoteObject?** **SF**

**A 53:** A remote object is one whose methods can be invoked from another JVM (or process). A remote object class must implement the *Remote* interface. A RMI Server is an application that creates a number of remote objects.

An RMI Server is responsible for

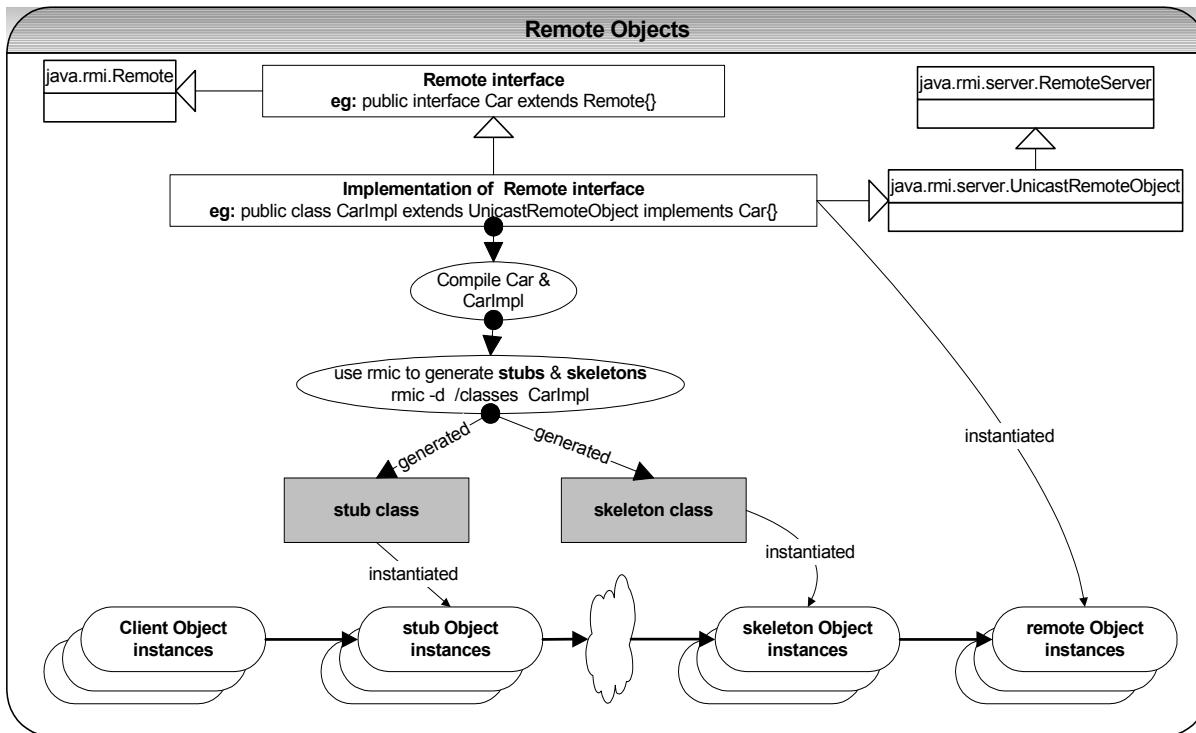
- Creating an instance of the remote object (e.g. `CarlImpl instance = new CarlImpl()`).
- **Exporting** the remote object.
- Binding the instance of the remote object to the RMI registry.

By exporting a remote object you make it available to accept incoming calls from the client. You can export the remote object by either extending the `java.rmi.server.UnicastRemoteObject` or if your class is already extending another class then you can use the static method

```
UnicastRemoteObject.exportObject (this);
```

If the UnicastRemoteObject is not extended (ie if you use UnicastRemoteObject.exportObject(...)) then the implementation class is responsible for the correct implementations of the hashCode(), equals() and toString() methods. A remote object is registered in the RMI registry using:

```
Naming.rebind(String serviceName, Remote remoteObj);
```



### Interoperability:

**Q 54:** What is the difference between RMI and CORBA? **SF**  
**A 54:** ability to integrate when we have diverse systems, platforms, languages, frameworks, applications

RMI	CORBA
Java only solution. The interfaces, implementations and the clients are all written in Java.	CORBA was made specifically for interoperability among various languages. For example the server could be written in C++ and the business logic can be in Java and the client can be written in COBOL.
RMI allows dynamic loading of classes at runtime.	In a CORBA environment with multi-language support it is not possible to have dynamic loading.

**Q 55:** What are the services provided by the RMI Object? **SF**

**A 55:** In addition to its remote object architecture RMI provides some basic object services, which can be used in a distributed application. These services are

- **Object naming/registry service:** RMI servers can provide services to clients by registering one or more remote objects with its local RMI registry.
- **Object activation service:** It provides a way for server (or remote) objects to be started on an as-needed basis. Without the remote activation service, a server object has to be registered with the RMI registry service.
- **Distributed garbage collection:** It is an automatic process where an object, which has no further remote references, becomes a candidate for garbage collection.

**Q 56:** What are the differences between RMI and a socket? **SF**

**A 56:**

Socket	RMI
A socket is a transport mechanism. Sockets are like applying procedural networking to object oriented environment.	RMI uses sockets. RMI is object oriented. Methods can be invoked on the remote objects running on a separate JVM.
Sockets-based network programming can be laborious.	RMI provides a convenient abstraction over raw sockets. Can send and receive any valid Java object utilizing underlying object serialization without having to worry about using data streams.

**Q 57:** How will you pass parameters in RMI? **SF**

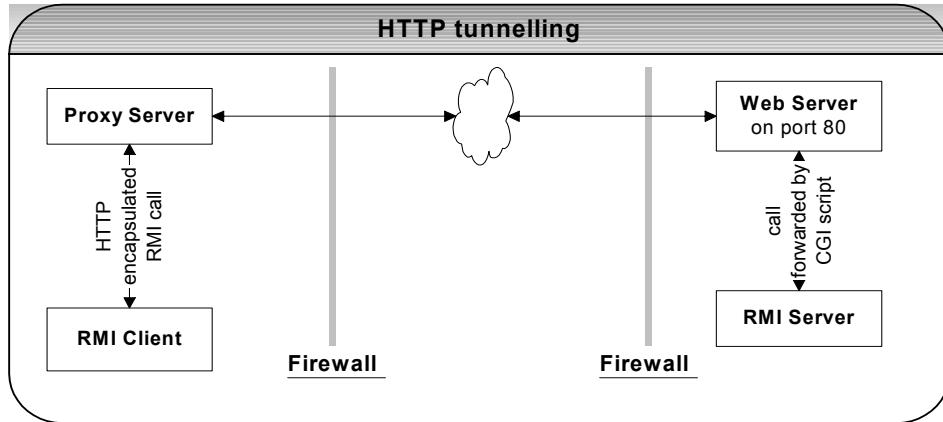
**A 57:**

- Primitive types are passed by value (e.g. int, char, boolean etc).
- References to remote objects (i.e. **objects which implements the Remote interface**) are passed as remote references that allows the client process to invoke methods on the remote objects.
- Non-remote objects are passed by value using object serialization. These objects should allow them to be serialized by implementing the `java.io.Serializable` interface.

**Note:** The client process initiates the invocation of the remote method by calling the method on the stub. The stub (client side proxy of the remote object) has a reference to the remote object and forwards the call to the skeleton (server side proxy of the remote object) through the reference manager by **marshalling** the method arguments. During marshalling each object is checked to determine whether it implements `java.rmi.Remote` interface. If it does then the remote reference is used as the marshalled data otherwise the object is serialized into byte streams and sent to the remote process where it is deserialized into a copy of the local object. The skeleton converts this request from the stub into the appropriate method call on the actual remote object by **unmarshalling** the method arguments into local stubs on the server (if they are remote reference) or into local copy (if they are sent as serialized objects).

**Q 58:** What is HTTP tunnelling or how do you make RMI calls across firewalls? **SF SE**

**A 58:** RMI transport layer generally opens direct sockets to the server. Many Intranets have firewalls that do not allow this. To get through the firewall an RMI call can be embedded within the firewall-trusted HTTP protocol. To get across firewalls, RMI makes use of **HTTP tunnelling** by encapsulating RMI calls within an HTTP POST request.



**When a firewall proxy server can forward HTTP requests only to a well-known HTTP port:** The firewall proxy server will forward the request to a HTTP server listening on port 80, and a CGI script will be executed to forward the call to the target RMI server port on the same machine.

**When a firewall proxy server can forward HTTP requests to any arbitrary port:** The firewall proxy will forward to any arbitrary port on the host machine and then it is forwarded directly to the port on which RMI Server is listening.

The disadvantages of HTTP tunnelling are performance degradation, prevents RMI applications from using callbacks, CGI script will redirect any incoming request to any port, which is a security loophole, RMI calls cannot be multiplexed through a single connection since HTTP tunnelling follows a request/response protocol etc.

**Q 59:** Why use RMI when we can achieve the same benefits from EJB? **SF**

**A 59:** EJBs are distributed components, which use the RMI framework for object distribution. An EJB application server provides more services like transactions, object pooling, database connection-pooling etc, which RMI does not provide. These extra services that are provided by the EJB server simplify the programming effort at the cost of performance overhead compared to plain RMI. So if performance is important then pure RMI may be a better solution (or under extreme situations Sockets can offer better performance than RMI).

**Note:** The decision to go for RMI or EJB or Sockets should be based on requirements such as maintainability, ease of coding, extensibility, performance, scalability, availability of application servers, business requirements etc.

### Enterprise – EJB 2.x

There are various persistence mechanisms available like EJB 2.x, Object-to-Relational (O/R) mapping tools like Hibernate, JDBC and EJB 3.0 (new kid on the block) etc. You will have to evaluate the products based on the application you are building because each product has its strengths and weaknesses. You will find yourself trading ease of use for scalability, standards with support for special features like stored procedures, etc. Some factors will be more important to you than for others. There is no one size fits all solution. Let's compare some of the persistence products:

EJB 2.x	EJB 3.0	Hibernate	JDBC
<b>PROS:</b> <ul style="list-style-type: none"> <li>▪ Security is provided for free for accessing the EJB.</li> <li>▪ Provides declarative transactions.</li> <li>▪ EJBs are pooled and cached. EJB life cycles are managed by the container.</li> <li>▪ Has remote access capabilities and can be clustered for scalability.</li> </ul> <b>Cons:</b> <ul style="list-style-type: none"> <li>▪ Need to understand the intricacies like rolling back a transaction, granularity etc, infrastructures like session facades, business delegates, value objects etc and strategies like lazy loading, dirty marker etc.</li> <li>▪ EJBs use lots of resources and have lots of artifacts.</li> <li>▪ Does not support OO concepts like inheritance.</li> </ul>	<b>PROS:</b> <ul style="list-style-type: none"> <li>▪ A lot less artefacts than EJB 2.x. Make use of annotations or attributes based programming.</li> <li>▪ Narrows the gap between EJB 2.x and O/R mapping.</li> <li>▪ Do support OO concepts like inheritance.</li> </ul> <b>Cons:</b> <ul style="list-style-type: none"> <li>▪ Since it is new, might be too early to use in commercial projects.</li> <li>▪ It is still evolving.</li> </ul>	<b>PROS:</b> <ul style="list-style-type: none"> <li>▪ Simple to write CRUD (create, retrieve, update, delete) operations.</li> <li>▪ No container or application server is required and can be plugged into an existing container.</li> <li>▪ Tools are available to simplify mapping relational data to objects and quick to develop.</li> </ul> <b>Cons:</b> <ul style="list-style-type: none"> <li>▪ Little or no capabilities for remote access and distributability.</li> <li>▪ Mapping schemas can be tedious and O/R mapping has its tricks like using lazy initialization, eager loading etc. What works for one may not work for another.</li> <li>▪ Limited clustering capabilities.</li> <li>▪ Large data sets can still cause memory issues.</li> <li>▪ Support for security at a database level only and no support for role based security without any add on APIs like Aspect Oriented Programming etc.</li> </ul>	<b>PROS:</b> <ul style="list-style-type: none"> <li>▪ You have complete control over the persistence because this is the building blocks of nearly all other persistence technologies in Java.</li> <li>▪ Can call Stored Procedures.</li> <li>▪ Can manipulate relatively large data sets.</li> </ul> <b>Cons:</b> <ul style="list-style-type: none"> <li>▪ You will have to write a lot of code to perform a little. Easy to make mistakes in properly managing connections and can cause out of cursors issues.</li> <li>▪ Harder to maintain because changes in schemas can cause lot of changes to your code.</li> <li>▪ Records need to be locked manually (e.g. select for update).</li> </ul>
As a rule of thumb, suitable for distributed and clustered applications, which is heavily transaction based. Records in use say between 1 and 50.	As a rule of thumb, suitable for distributed and clustered applications, which is heavily transaction based. Records in use say between 1 and 100.	Suitable for records in use between 100 and 5000. Watch out for memory issues, when using large data sets.	Where possible stay away from using JDBC unless you have compelling reason to use it for batch jobs where large amount of data need to be transferred, records in use greater than 5000, required to use Stored Procedures etc.

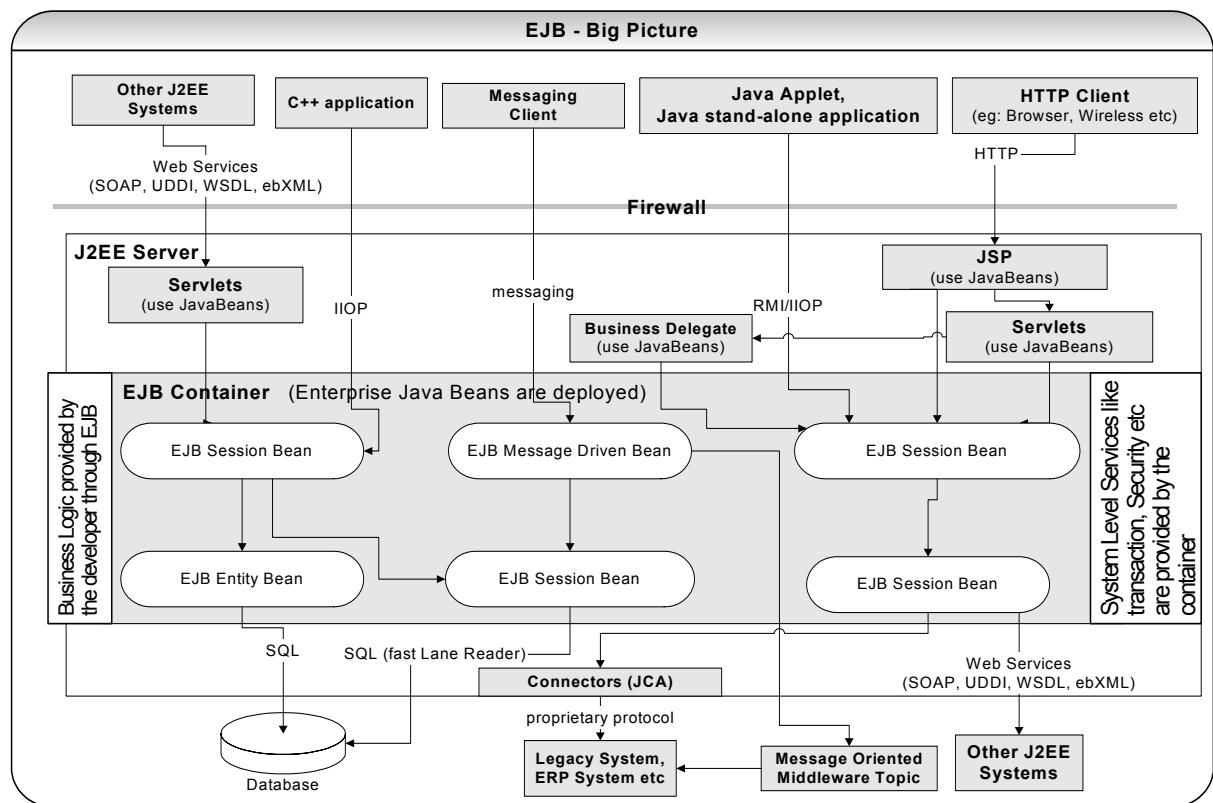
The stateless session beans and message driven beans have wider acceptance in EJB 2.x compared to stateful session beans and entity beans. Refer Emerging Technologies/Frameworks section for Hibernate and EJB 3.0.

**Q 60:** What is the role of EJB 2.x in J2EE? **SF**

**A 60:** EJB 2.x (Enterprise JavaBeans) is a widely adopted server side component architecture for J2EE.

- EJB is a remote, distributed multi-tier system and supports protocols like JRMP, IIOP, and HTTP etc.
- It enables rapid development of reusable, versatile, portable business components across middleware, transactional and scalable applications.
- EJB is a specification for J2EE servers. EJB components contain only business logic and system level programming and services like transactions, security, instance pooling, threading, persistence etc are managed by the EJB Container and hence simplify the programming effort.
- Message driven EJBs have support for asynchronous communication.

**Note:** Having said that EJB 2.x is a widely adopted server side component, **EJB 3.0** is taking ease of development very seriously and has adjusted its model to offer the POJO (Plain Old Java Object) persistence and the new **O/R mapping model based on Hibernate**. In EJB 3.0, **all kinds of enterprise beans are just POJOs**. EJB 3.0 **extensively uses Java annotations**, which replaces excessive XML, based configuration files and eliminates the need for the rigid component model used in EJB 1.x, 2.x. Annotations can be used to define the bean's business interface, O/R mapping information, resource references etc. Refer **Q18** in Emerging Technologies/Frameworks section. So, for future developments look out for EJB 3.0 and/or Hibernate framework. Refer **Q14 – Q16** in Emerging Technologies/Frameworks section for discussion on Hibernate framework.



**Q 61:** What is the difference between EJB and JavaBeans? **SF**

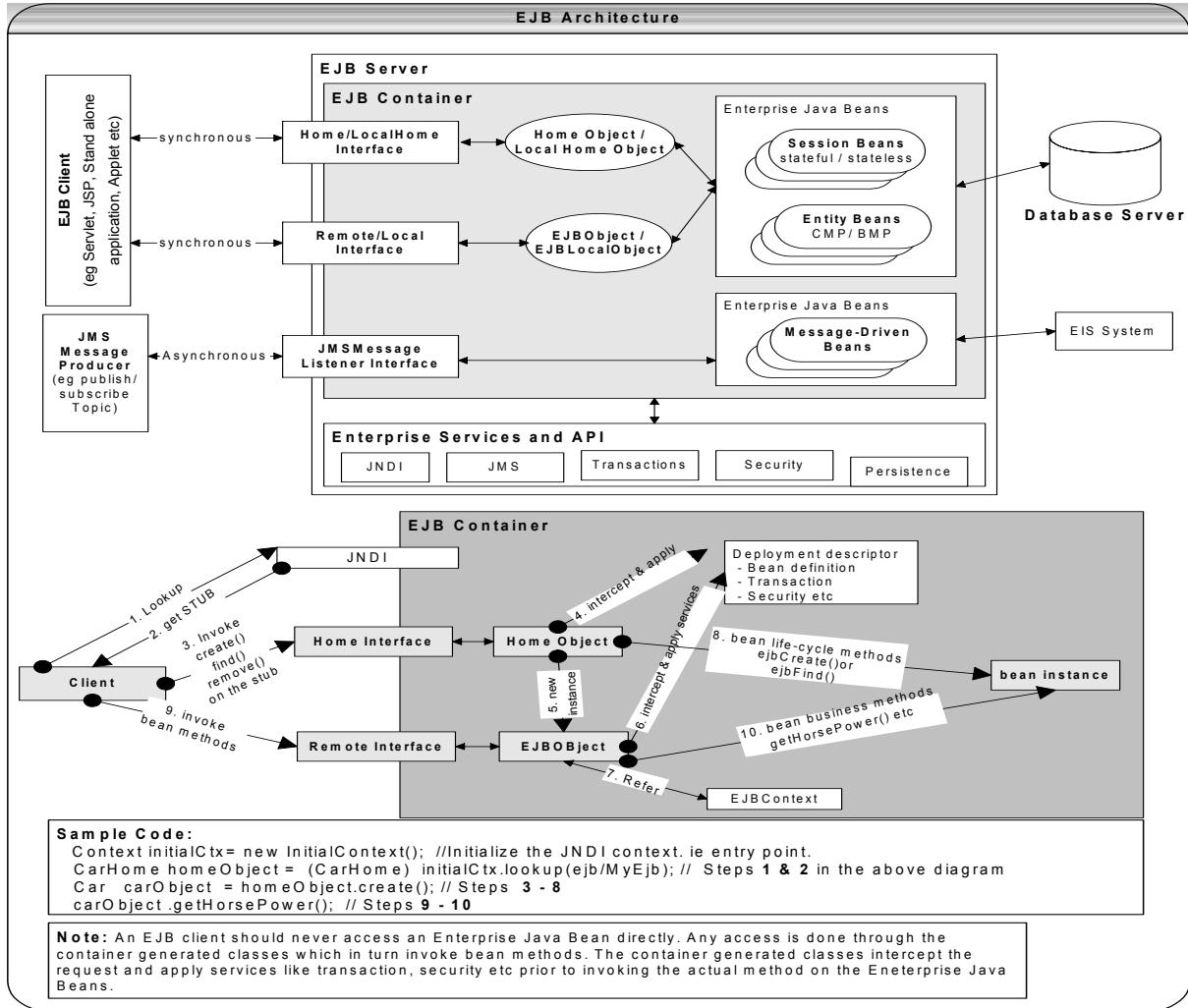
**A 61:** Both EJB and JavaBeans have very similar names but this is where the similarities end.

JavaBeans	Enterprise JavaBeans (EJB)
The components built based on JavaBeans live in a single local JVM (or address space) and can be either visual (e.g. GUI components like Button, List etc) or non-visual at runtime.	The Enterprise JavaBeans are non-visual distributable components, which can live across multiple JVMs (or address spaces).

No explicit support exists for services like transactions etc.	EJBs can be transactional and the EJB servers provide transactional support.
JavaBeans are fine-grained components, which can be used to assemble coarse-grained components or an application.	EJBs are coarse-grained components that can be deployed as is or assembled with other components into larger applications. EJBs must be deployed in a container that provides services like instance pooling, multi-threading, security, life-cycle management, transactions etc
Must conform to JavaBeans specification.	Must conform to EJB specification.

**Q 62:** Explain EJB architecture? **SF**

**A 62:**



**EJB Container:** EJBs are software components, which run in an environment called an EJB container. An EJB cannot function outside an EJB Container. The EJB container hosts and manages an Enterprise JavaBean in a similar manner that a Web container hosts a servlet or a Web browser hosts a Java Applet. The EJB container manages the following services so that the developer can concentrate on writing the business logic:

- Transactions (refer Q71 – Q75 in Enterprise section)
- Persistence
- EJB instance pooling
- Security (refer Q81 in Enterprise section)
- Concurrent access (or multi-threading)
- Remote access

**Design pattern:** EJBs use the proxy design pattern to make remote invocation (i.e. remote proxy) and to add container managed services like security and transaction demarcation. Refer Q11 in “How would you about...” section for a more detailed discussion on proxy design pattern and dynamic proxies.

**EJBContext:** Every bean obtains an EJBContext object, which is a reference directly to the container. The EJB can request information about its environment like the status of a transaction, a remote reference to itself (an EJB cannot use 'this' to reference itself) etc.

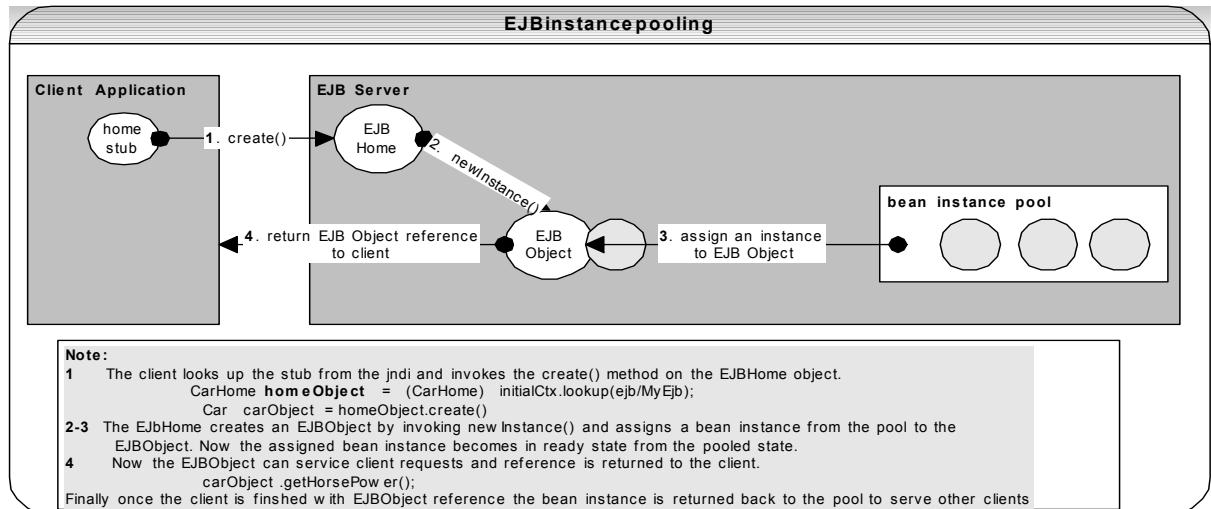
**Deployment Descriptor:** The container handles all the above mentioned services declaratively for an EJB based on the XML deployment descriptor (`ejb-jar.xml`). When an EJB is deployed into a container the deployment descriptor is read to find out how these services are handled. Refer to the *J2EE deployment structure* diagram in **Q6** in Enterprise section.

**EJB:** The EJB architecture defines 3 distinct types of Enterprise JavaBeans.

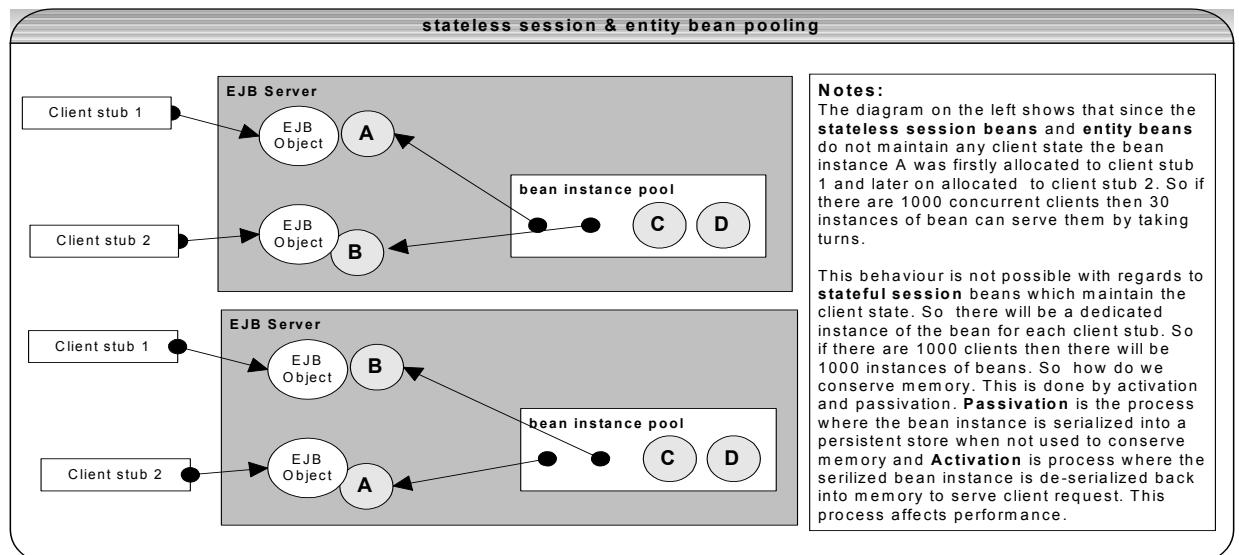
- Session beans.
- Entity beans.
- Message-driven beans.

The session and entity beans are invoked synchronously by the client and message driven beans are invoked asynchronously by a message container such as a publish/subscribe topic. Let's look at some of the EJB container services in a bit more detail:

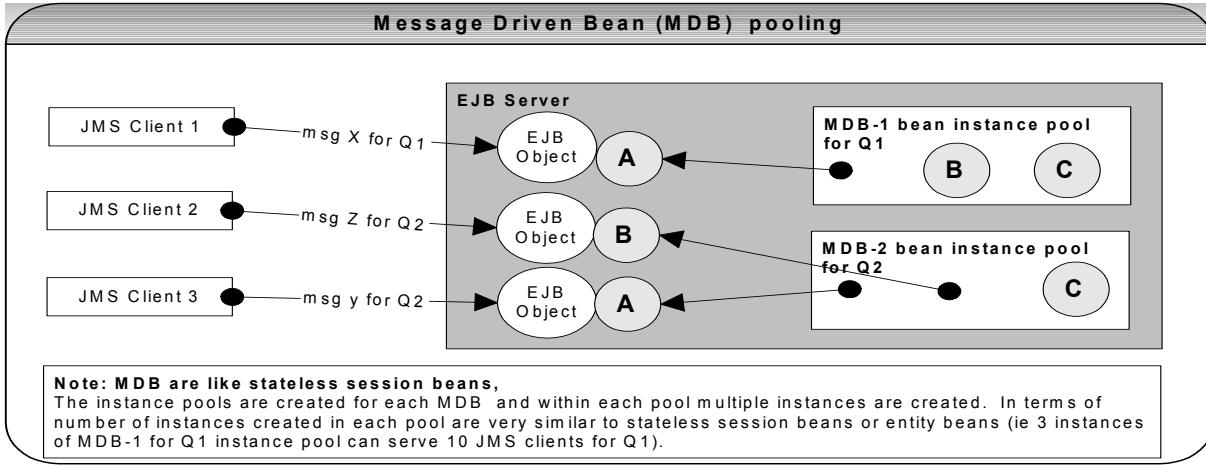
### Instance pooling



The above diagram shows how the EJB instances are pooled and assigned to EJB Object and then returned to the pool. Let's look at in detail for different types of EJBs.



From the diagrams it is clear that bean instances can be reused for all the bean types except for the stateful session bean where the client state is maintained. So we need a dedicated stateful session bean for each client.



### Concurrent access

The session beans do not support concurrent access. The stateful session beans are exclusively for a client so there is no concurrent access. The stateless session beans do not maintain any state. It does not make any sense to have concurrent access. The Entity beans represent data that is in the database table, which is shared between the clients. So to make concurrent access possible the EJB container need to protect the data while allowing many clients simultaneous access. When you try to share distributed objects you may have the following problem:

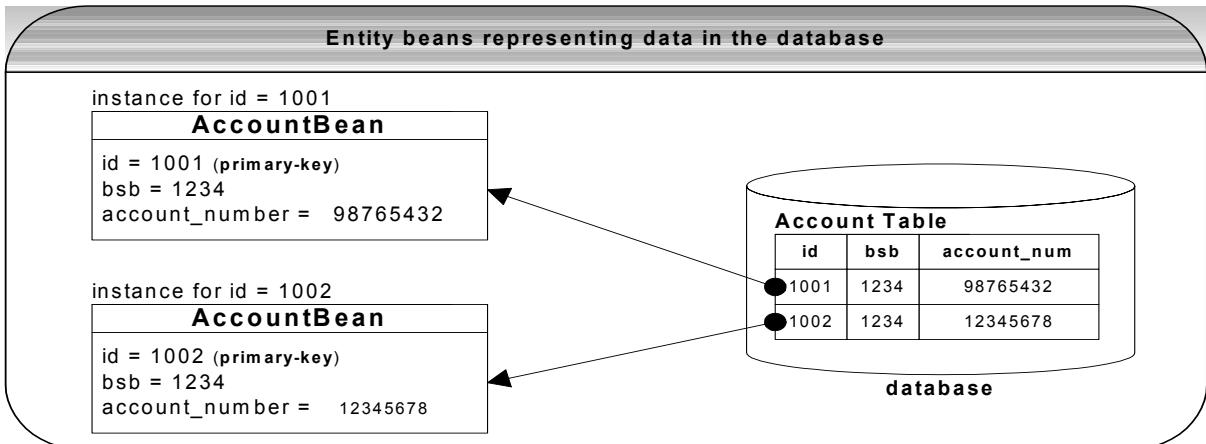
If 2 clients are using the same EJBObject, how do you keep one client from writing over the changes of the other? Say for example

Client-1 reads a value x= 5  
Client-2 modifies the value to x=7  
Now the client-1's value is invalid.

The entity bean addresses this by prohibiting concurrent access to bean instances. Which means several clients can be connected to one EJBObject but only one client can access the EJB instance at a time.

### Persistence

Entity beans basically represent the data in a relational database. An Entity Bean is responsible for keeping its state in sync with the database.



- Container-managed persistence (CMP) - The container is responsible for saving the bean's state with the help of object-relational mapping tools.
- Bean-managed persistence (BMP) – The Entity Bean is responsible for saving its own state.

If entity beans performance is of concern then there are other persistence technologies and frameworks like JDBC, JDO, Hibernate, OJB and Oracle TopLink (commercial product).

---

**Q 63:** What are the different kinds of enterprise beans? **SF**

**A 63:**

**Session Bean:** is a non-persistent object that implements some business logic running on the server. Session beans do not survive system shut down. There are two types of session beans

- Stateless session beans (each session bean can be reused by multiple EJB clients)
- Stateful session beans (each session bean is associated with one EJB client)

**Entity Bean:** is a persistent object that represents object views of the data, usually a row in a database. They have the primary key as a unique identifier. Multiple EJB clients can share each entity bean. Entity beans can survive system shut downs. Entity beans can have two types of persistence

- Container-managed persistence (CMP) - The container is responsible for saving the bean's state.
- Bean-managed persistence (BMP) – The Entity Bean is responsible for saving its own state.

**Message-driven Bean:** is integrated with the Java Message Service (JMS) to provide the ability to act as a message consumer and perform asynchronous processing between the server and the message producer.

---

**Q 64:** What is the difference between session and entity beans? **SF**

**A 64:**

Session Beans	Entity Beans
Use session beans for application logic.	Use entity beans to develop persistent object model.
Expect little reuse of session beans.	Insist on reuse of entity beans.
Session beans control the workflow and transactions of a group of entity beans.	Domain objects with a unique identity (ie-primary key) shared by multiple clients.
Life is limited to the life of a particular client. Handle database access for a particular client.	Persist across multiple invocations. Handles database access for multiple clients.
Do not survive system shut downs or server crashes.	Do survive system shut downs or server crashes.

---

**Q 65:** What is the difference between stateful and stateless session beans? **SF**

**A 65:**

Stateless Session Beans	Stateful Session Bean
Do not have an internal state. Can be reused by different clients.	Do have an internal state. Reused by the same client.
Need not be activated or passivated since the beans are pooled and reused.	Need to handle activation and passivation to conserve system memory since one session bean object per client.

---

**Q 66:** What is the difference between Container Managed Persistence (CMP) and Bean Managed Persistence (BMP)? **SF**

**A 66:**

Container Managed Persistence (CMP)	Bean Managed Persistence (BMP)
The container is responsible for persisting state of the bean.	The bean is itself responsible for persisting its own state.
Container needs to generate database (SQL) calls.	The bean needs to code its own database (SQL) calls.
The bean persistence is independent of its database (e.g. DB2, Oracle, Sybase etc). So it is portable from one data source to another.	The bean persistence is hard coded and hence may not be portable between different databases (e.g. DB2, Oracle etc).

---

**Q 67:** Can an EJB client invoke a method on a bean directly? **SF**

**A 67:** An EJB client should never access an EJB directly. Any access is done through the container. The container will intercept the client call and apply services like transaction, security etc prior to invoking the actual EJB. This relationship between the EJB and the container is like “**don't call us, we will call you**”.

---

**Q 68:** How does an EJB interact with its container and what are the call-back methods in entity beans? **SF**

**A 68:** EJB interacts with its container through the following mechanisms

- **Call-back Methods:** Every EJB implements an interface (extends EnterpriseBean) which defines several methods which alert the bean to various events in its lifecycle. A container is responsible for invoking these methods. These methods notify the bean when it is about to be activated, to be persisted to the database, to end a transaction, to remove the bean from the memory, etc. For example the entity bean has the following call-back methods:

```
public interface javax.ejb.EntityBean {

 public void setEntityContext(javax.ejb.EntityContext c);
 public void unsetEntityContext();
 public void ejbLoad();
 public void ejbStore();
 public void ejbActivate();
 public void ejbPassivate();
 public void ejbRemove();
}
```

- **EJBContext:** provides methods for interacting with the container so that the bean can request information about its environment like the identity of the caller, security, status of a transaction, obtains remote reference to itself etc. e.g. isUserInRole(), getUserPrincipal(), isRollbackOnly(), etc
- **JNDI (Java Naming and Directory Interface):** allows EJB to access resources like JDBC connections, JMS topics and queues, other EJBs etc.

**Q 69:** What is the difference between EJB 1.1 and EJB 2.0? What is the difference between EJB 2.x and EJB 3.0? **SF**

**A 69:** EJB 2.0 has the following additional advantages over the EJB 1.1

- **Local interfaces:** These are beans that can be used locally, that means by the same Java Virtual Machines, so they do not require to be wrapped like remote beans, and arguments between those interfaces are passed directly by reference instead of by value. This improves performance.
- **ejbHome methods:** Entity beans can declare ejbHome methods that perform operations related to the EJB component but that are not specific to a bean instance.
- **Message Driven Beans (MDB):** is a completely new enterprise bean type, which is designed specifically to handle incoming JMS messages.
- **New CMP Model.** It is based on a new contract called *the abstract persistence schema*, which will allow to the container to handle the persistence automatically at runtime.
- **EJB Query Language:** It is a sql-based language that will allow the new persistence schema to implement and execute finder methods.

Let's look at some of the new features on EJB 2.1

- **Container-managed timer service:** The timer service provides coarse-grained, transactional, time-based event notifications to enable enterprise beans to model and manage higher-level business processes.
- **Web service support:** EJB 2.1 adds the ability of stateless session beans to implement a Web service endpoint via a Web service endpoint interface.
- **EJB-QL:** Enhanced EJB-QL includes support for aggregate functions and ordering of results.

Current **EJB 2.x** model is complex for a variety of reasons:

- You need to create several component interfaces and implement several unnecessary call-back methods.
- EJB deployment descriptors are complex and error prone.
- EJB components are not truly object oriented, as they have restrictions for using inheritance and polymorphism.
- EJB modules cannot be tested outside an EJB container and debugging an EJB inside a container is very difficult.

**Note:** EJB 3.0 is taking ease of development very seriously and has adjusted its model to offer the POJO (Plain Old Java Object) persistence and the new **O/R mapping model based on Hibernate**. In EJB 3.0, all kinds of enterprise beans are just **POJOs**. EJB 3.0 **extensively uses Java annotations**, which replaces excessive XML based configuration files and eliminate the need for rigid component model used in EJB 1.x, 2.x. Annotations can be used to define the bean's business interface, O/R mapping information, resource references etc. Refer Q18 in Emerging Technologies/Frameworks section.

**Q 70:** What are the implicit services provided by an EJB container? **SF**

**A 70:**

- **Lifecycle Management:** Individual enterprise beans do not need to explicitly manage process allocation, thread management, object activation, or object destruction. The EJB container automatically manages the object lifecycle on behalf of the enterprise bean.
- **State Management:** Individual enterprise beans do not need to explicitly save or restore conversational object state between method calls. The EJB container automatically manages object state on behalf of the enterprise bean.
- **Security:** Individual enterprise beans do not need to explicitly authenticate users or check authorisation levels. The EJB container automatically performs all security checking on behalf of the enterprise bean.
- **Transactions:** Individual enterprise beans do not need to explicitly specify transaction demarcation code to participate in distributed transactions. The EJB container can automatically manage the start, enrolment, commitment, and rollback of transactions on behalf of the enterprise bean.
- **Persistence:** Individual enterprise beans do not need to explicitly retrieve or store persistent object data from a database. The EJB container can automatically manage persistent data on behalf of the enterprise bean.

**Q 71:** What are transactional attributes? **SF TI**

**A 71:** EJB transactions are a set of mechanisms and concepts, which insures the integrity and consistency of the database when multiple clients try to read/update the database simultaneously.

**Transaction attributes** are defined at different levels like EJB (or class), a method within a class or segment of a code within a method. The attributes specified for a particular method take precedence over the attributes specified for a particular EJB (or class). Transaction attributes are specified *declaratively* through EJB deployment descriptors. Unless there is any compelling reason, the *declarative* approach is recommended over programmatic approach where all the transactions are handled programmatically. With the *declarative* approach, the EJB container will handle the transactions.

Transaction Attributes	Description
Required	Methods executed within a transaction. If client provides a transaction, it is used. If not, a new transaction is generated. Commit at end of method that started the transaction. Which means a method that has <i>Required</i> attribute set, but was called when the transaction has already started will not commit at the method completion. Well suited for EJB session beans.
Mandatory	Client of this EJB must create a transaction in which this method operates, otherwise an error will be reported. Well-suited for entity beans.
RequiresNew	Methods executed within a transaction. If client provides a transaction, it is suspended. If not a new transaction is generated, regardless. Commit at end of method.
Supports	Transactions are optional.
NotSupported	Transactions are not supported. If provided, ignored.
Never	Code in the EJB responsible for explicit transaction control.

**Q 72:** What are isolation levels? **SF TI PI**

**A 72:** Isolation levels provide a degree of control of the effects one transaction can have on another concurrent transaction. Since concurrent effects are determined by the precise ways in which, a particular relational database handles locks and its drivers may handle these locks differently. The semantics of isolation mechanisms based on these are not well defined. Nevertheless, certain defined or approximate properties can be specified as follows:

Isolation level	Description
TRANSACTION_SERIALIZABLE	Strongest level of isolation. Places a range lock on the data set, preventing other users from updating or inserting rows into the data set until the transaction is complete. Can produce deadlocks.

**Non-Repeatable Read - In a same transaction same query yields different results.** This happens when another transaction updates the data returned by other transaction. Enterprise Java

**Phantom Read - First txn still not done adding data, and may add more data after second txn made a read**

TRANSACTION_REPEATABLE_READ	Locks are placed on all data that is used in a query, preventing other users from updating the data, but new <b>phantom records</b> can be inserted into the data set by another user and are included in later reads in the current transaction.
TRANSACTION_READ_COMMITTED	Can't read uncommitted data by another transaction. Shared locks are held while the data is being read to avoid <b>dirty reads</b> , but the data can be changed before the end of the transaction resulting in <b>non-repeatable reads</b> and <b>phantom records</b> .
TRANSACTION_READ_UNCOMMITTED	Can read uncommitted data ( <b>dirty read</b> ) by another transaction, and <b>non-repeatable reads</b> and <b>phantom records</b> are possible. Least restrictive of all isolation levels. No shared locks are issued and no exclusive locks are honoured.

Isolation levels are not part of the EJB specification. They can only be set on the resource manager either explicitly on the *Connection* (for bean managed persistence) or via the application server specific configuration. The EJB specification indicates that isolation level is part of the **Resource Manager**.

As the transaction **isolation level increases**, likely **performance degradation** follows, as additional locks are required to protect data integrity. If the underlying data does not require such a high degree of integrity, the isolation level can be lowered to improve performance.

**Q 73:** What is a distributed transaction? What is a 2-phase commit? SF TI

**A 73:** A **Transaction** (Refer Q43 in Enterprise section) is a series of actions performed as a single unit of work in which either all of the actions performed as a logical unit of work in which, either all of the actions are performed or none of the actions. A transaction is often described by ACID properties (Atomic, Consistent, Isolated and Durable). A **distributed transaction** is an ACID transaction between two or more independent transactional resources like two separate databases. For the transaction to commit successfully, all of the individual resources must commit successfully. If any of them are unsuccessful, the transaction must rollback in all of the resources. A **2-phase commit** is an approach for committing a distributed transaction in 2 phases.

**Phase 1 is prepare:** Each of the resources votes on whether it's ready to commit – usually by going ahead and persisting the new data but not yet deleting the old data.

**Phase 2 is committing:** If all the resources are ready, they all commit – after which old data is deleted and transaction can no longer roll back. 2-phase commit ensures that a distributed transaction can always be committed or always rolled back if one of the databases crashes. The **XA** specification defines how an application program uses a transaction manager to coordinate distributed transactions across multiple resource managers. Any resource manager that adheres to XA specification can participate in a transaction coordinated by an XA-compliant transaction manager.

**Q 74:** What is dooming a transaction? TI

**A 74:** A transaction can be doomed by the following method call CO

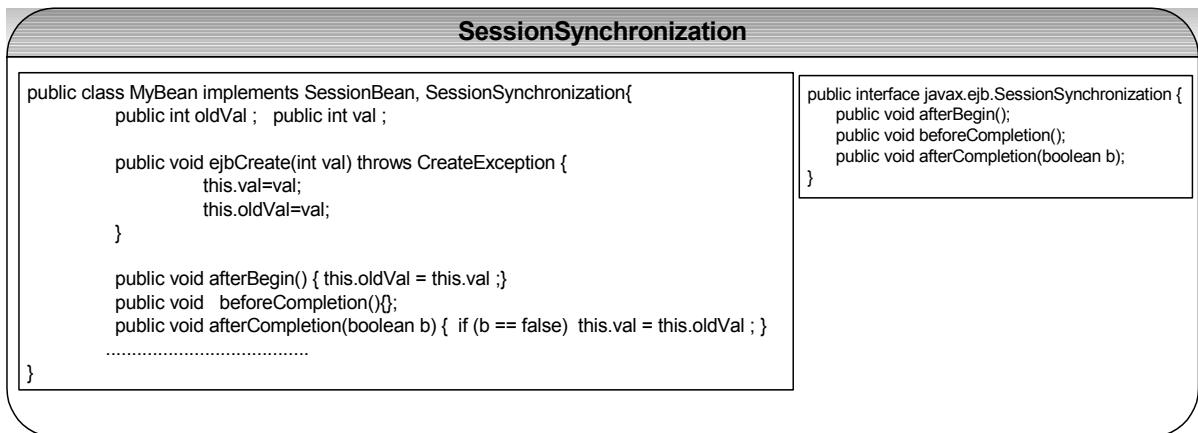
```
EJBContext.setRollbackOnly();
```

The above call will force transaction to rollback. The doomed transactions decrease scalability and if a transaction is doomed why perform compute intensive operations? So we can detect a doomed transaction as shown below: CO

```
public void doComputeIntensiveOperation() throws Exception {
 if (ejbContext.getRollbackOnly()) {
 return; //transaction is doomed so return (why unnecessarily perform compute intensive operation)
 }
 else {
 performComplexOperation();
 }
}
```

**Q 75:** How to design transactional conversations with session beans? SF TI

**A 75:** A stateful session bean is a resource which has an in memory state which can be rolled back in case of any failure. It can participate in transactions by implementing SessionSynchronization. CO



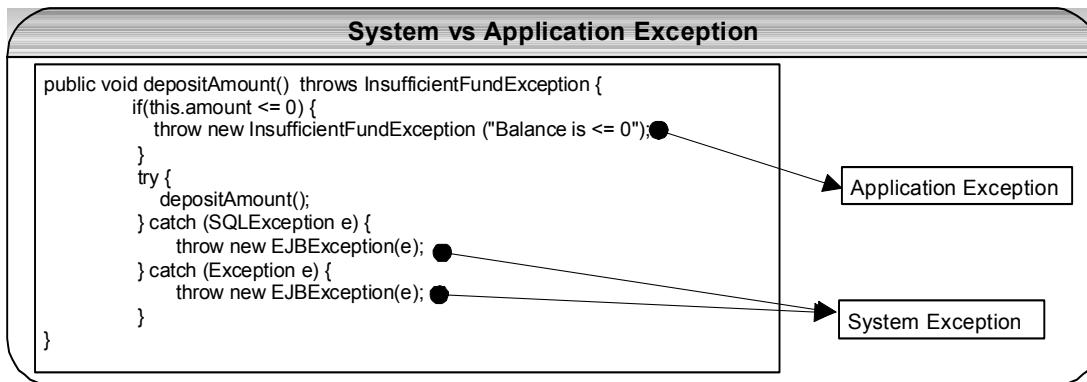
The uses of SessionSynchronization are:

- Enables the bean to act as a transactional resource and undo state changes on failure.
- Enables you to cache database data to improve performance.

**Q 76:** Explain exception handling in EJB? **SF EH CO**

**A 76:** Java has two types of exceptions:

- **Checked exception:** derived from **java.lang.Exception** but not **java.lang.RuntimeException**.
- **Unchecked exception:** derived from **java.lang.RuntimeException** thrown by JVM.



EJB has two types of exceptions:

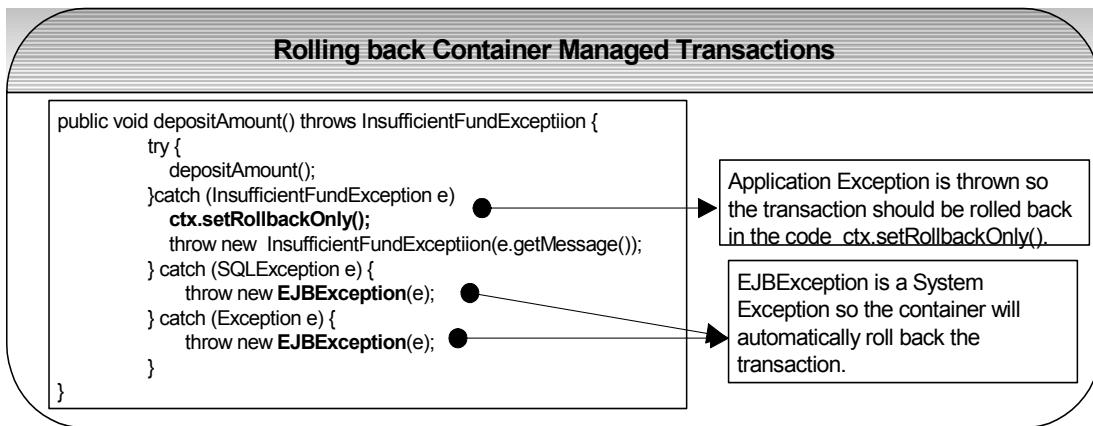
- **System Exception:** is an **unchecked** exception derived from **java.lang.RuntimeException**.
- **Application Exception:** is specific to an application and thrown because of violation of business rules.

A **System Exception** is thrown by the system and is not recoverable. For example EJB container losing connection to the database server, failed remote method objects call etc. Because the System Exceptions are unpredictable, the EJB container is the only one responsible for trapping the System Exceptions. The container automatically wraps any **RuntimeException** in **RemoteException**, which subsequently gets thrown to the caller (or client). In addition to intercepting System Exception the container may log the errors.

An **Application Exception** is specific to an application and is thrown because of violation of business rules. The client should be able to determine how to handle an Application Exception. If the account balance is zero then an Application Exception like **InsufficientFundException** can be thrown. If an **Application Exception** should be treated as a System Exception (e.g. SQLException) it needs to be wrapped in an **EJBException** so that it can be managed properly and propagated to the client.

**Q 77:** How do you rollback a container managed transaction in EJB? **SF TI EH**

**A 77:** The way the exceptions are handled affects the way the transactions are managed. 



When the container manages the transaction, it is automatically rolled back when a **System Exception** occurs. This is possible because the container can intercept System Exception. However when an **Application Exception** occurs, the container does not intercept it and therefore leaves it to the code to roll back using `ctx.setRollbackOnly()`.

Be aware that handling exceptions in EJB is different from handling exceptions in Java. The Exception handling best practice tips are:

- If you cannot recover from System Exception let the container handle it.
- If a business rule is violated then throw an application exception.
- Catch the Exceptions in a proper order.
- It is a poor practice to catch `java.lang.Exception` because this is a big basket, which will catch all the unhandled exceptions. It is shown in the above diagrams for illustration purpose only. You should avoid this because if you add a new piece of code, which throws a new, checked exception, then the compiler won't pick it up.

**Q 78:** What is the difference between optimistic and pessimistic concurrency control? 

**A 78:**

Pessimistic Concurrency	Optimistic Concurrency
A pessimistic design assumes conflicts will occur in the database tables and avoids them through exclusive locks etc.	An optimistic approach assumes conflicts won't occur, and deal with them when they do occur.
EJB (also non-EJB) locks the source data until it completes its transaction. <ul style="list-style-type: none"> <li>▪ Provides reliable access to data.</li> <li>▪ Suitable for short transactions.</li> <li>▪ Suitable for systems where concurrent access is rare.</li> </ul>	EJB (also non-EJB) implements a strategy to detect whether a change has occurred. Locks are placed on the database only for a small portion of the time. <ul style="list-style-type: none"> <li>▪ Suitable for long transactions.</li> <li>▪ Suitable for systems requiring frequent concurrent accesses.</li> </ul>
The pessimistic locking imposes high locking overheads on the server and lower concurrency.	The optimistic locking is used in the context of cursors. The optimistic locking works as follows: <ul style="list-style-type: none"> <li>▪ No locks are acquired as rows are read.</li> <li>▪ No locks are acquired while values in the current row are changed.</li> <li>▪ When changes are saved, a copy of the row in the database is read in the locked mode.</li> <li>▪ If the data was changed after it was read into the cursor, an error is raised so that the transaction can be rolled back and retried.</li> </ul> <p><b>Note:</b> The testing for changes can be done by comparing the values, timestamp or version numbers.</p>

**Q 79:** How can we determine if the data is stale (for example when using optimistic locking)? 

**A 79:** We can use the following strategy to determine if the data is stale:

- Adding version numbers

1. Add a version number (Integer) to the underlying table.
2. Carry the version number along with any data read into memory (through value object, entity bean etc).
3. Before performing any update compare the current version number with the database version number.
4. If the version numbers are equal update the data and increment the version number.
5. If the value object or entity bean is carrying an older version number, reject the update and throw an exception.

**Note:** You can also do the version number check as part of the update by including the version column in the where clause of the update without doing a prior select.

- Adding a timestamp to the underlying database table.
- Comparing the data values.

These techniques are also quite useful when implementing data caching to improve performance. Data caches should regularly keep track of stale data to refresh the cache. These strategies are valid whether you use EJB or other persistence mechanisms like JDBC, Hibernate etc.

---

**Q 80:** What are not allowed within the EJB container? **SF**

**A 80:** In order to develop reliable and portable EJB components, the following restrictions apply to EJB code implementation:

- Avoid using static non-final fields. Declaring all static fields in EJB component as final is recommended. This enables the EJB container to distribute instances across multiple JVMs.
- Avoid starting a new thread (conflicts with EJB container) or using thread synchronization (allow the EJB container to distribute instances across multiple JVMs).
- Avoid using AWT or Swing functionality. EJBs are server side business components.
- Avoid using file access/java.io operations. EJB business components are meant to use resource managers such as JDBC to store and retrieve application data. Also deployment descriptors can be used to store <env-entry>.
- Avoid accepting or listening to socket connections. EJB components are not meant to provide network socket functionality. However the specification lets EJB components act as socket clients or RMI clients.
- Avoid using the reflection API. This restriction enforces Java security.
- Can't use custom class loaders.

---

**Q 81:** Discuss EJB container security? **SF SE**

**A 81:** EJB components operate inside a container environment and rely heavily on the container to provide security. The four key services required for the security are:

- **Identification:** In Java security APIs this identifier is known as a **principal**.
- **Authentication:** To prove the identity one must present the credentials in the form of password, swipe card, digital certificate, finger prints etc.
- **Authorisation (Access Control):** Every secure system should limit access to particular users. The common way to enforce access control is by maintaining **security roles** and **privileges**.
- **Data Confidentiality:** This is maintained by encryption of some sort. It is no good to protect your data by authentication if someone can read the password.

The EJB specification concerns itself exclusively with **authorisation** (access control). An application using EJB can specify in an abstract (declarative) and portable way that is allowed to access business methods. The EJB container handles the following actions:

- Find out the Identity of the caller of a business method.

- Check the EJB deployment descriptor to see if the identity is a member of a security role that has been granted the right to call this business method.
- Throw java.rmi.RemoteException if the access is illegal.
- Make the identity and the security role information available for a fine grained programmatic security check.

```
public void closeAccount() {
 if (ejbContext.getCallerPrincipal().getName() = "SMITH") {
 ...
 }

 if (!ejbContext.isCallerInRole(CORPORATE_ACCOUNT_MANAGER)) {
 throw new SecurityException("Not authorized to close this account");
 }
}
```

- Optionally log any illegal access.

There are two types of information the EJB developer has to provide through the deployment descriptor.

- Security roles
- Method permissions

**Example:**

```
<security-role>
<description>
 Allowed to open and close accounts
</description>
<role-name>account_manager</role-name>
</security-role>
<security-role>
<description>
 Allowed to read only
</description>
<role-name>teller</role-name>
</security-role>
```

There is a many-to-many relationship between the security roles and the method permissions.

```
<method-permission>
<role-name>teller</role-name>
<method>
<ejb-name>AccountProcessor</ejb-name>
<method-name>findByPrimaryKey</method-name>
</method>
</method-permission>
```

Just as we must declare the resources accessed in our code for other EJBs that we reference in our code we should also declare the security role we access programmatically to have a fine grained control as shown below.

```
<security-role-ref>
<description>
 Allowed to open and close accounts
</description>
<role-name>account_manager</role-name>
<role-link>executive</role-link>
</security-role-ref>
```

There is also many-to-many relationship between the EJB specific security roles that are in the deployment descriptor and the application based target security system like LDAP etc. For example there might be more than one group users and individual users that need to be mapped to a particular EJB security role 'account\_manager'.

**Q 82:** What are EJB best practices? **BP**

**A 82:**

- Use local interfaces that are available in EJB2.0 if you deploy both the EJB client and the EJB in the same server. Use vendor specific pass-by-reference implementation to make EJB1.1 remote EJBs operate as local.

[Extreme care should be taken not to affect the functionality by switching the application, which was written and tested in pass-by-reference mode to pass-by-value without analysing the implications and re-testing the functionality.

- Wrap entity beans with session beans to reduce network calls (refer **Q84** in Enterprise section) and promote declarative transactions. Where possible use local entity beans and session beans can be either local or remote. Apply the appropriate EJB design patterns as described in **Q83 – Q87** in Enterprise section.
- Cache ejbHome references to avoid JNDI look-up overhead using service locator pattern.
- Handle exceptions appropriately (refer **Q76, Q77** in Enterprise section).
- Avoid transaction overhead for non-transactional methods of session beans by declaring transactional attribute as 'Supports'.
- Choose plain Java object over EJB if you do not want services like RMI/IOP, transactions, security, persistence, thread safety etc. There are alternative frameworks such as Hibernate, Spring etc.
- Choose Servlet's **HttpSession** object rather than stateful session bean to maintain client state if you do not require component architecture of a stateful bean.
- Apply **Lazy loading** and **Dirty marker** strategies as described in **Q88** in Enterprise section.

Session Bean (stateless)	Session Bean (stateful)	Entity Bean
<ul style="list-style-type: none"> <li>▪ Tune the pool size to avoid overhead of creation and destruction.</li> <li>▪ Use setSessionContext(..) or ejbCreate(..) method to cache any bean specific resources.</li> <li>▪ Release any acquired resources like Database connection etc in ejbRemove() method</li> </ul>	<ul style="list-style-type: none"> <li>▪ Tune the pool size to avoid overhead of creation and destruction.</li> <li>▪ Set proper time out to avoid resource congestion.</li> <li>▪ Remove it explicitly from client using remove() method.</li> <li>▪ Use 'transient' variable where possible to avoid serialization overhead.</li> </ul>	<ul style="list-style-type: none"> <li>▪ Tune the pool size to avoid overhead of creation and destruction.</li> <li>▪ Use setEntityContext(..) method to cache any bean specific resources and unsetEntityContext() method to release acquired resources.</li> <li>▪ Use lazy-loading to avoid any unnecessary loading of dependent data. Use dirty marker to avoid unchanged data update.</li> <li>▪ Commit the data after a transaction completes to reduce any database calls in between.</li> <li>▪ Where possible perform bulk updates, use CMP rather than BMP. Use direct JDBC (Fast-lane-reader) instead of entity beans, use of read-only entity beans etc.</li> </ul>

**Q 83:** What is a business delegate? Why should you use a business delegate? **DP PI**

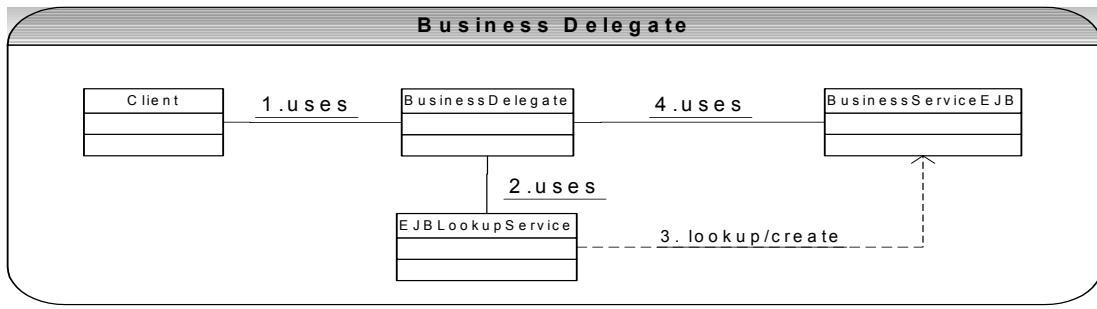
**A 83:** Questions **Q83 – Q88** are very popular EJB questions.

**Problem:** When presentation tier components interact directly with the business services components like EJB, the presentation components are vulnerable to changes in the implementation of business services components.

**Solution:** Use a **Business Delegate** to reduce the coupling between the presentation tier components and the business services tier components. Business Delegate hides the underlying implementation details of the business service, such as look-up and access details of the EJB architecture.

Business delegate is responsible for:

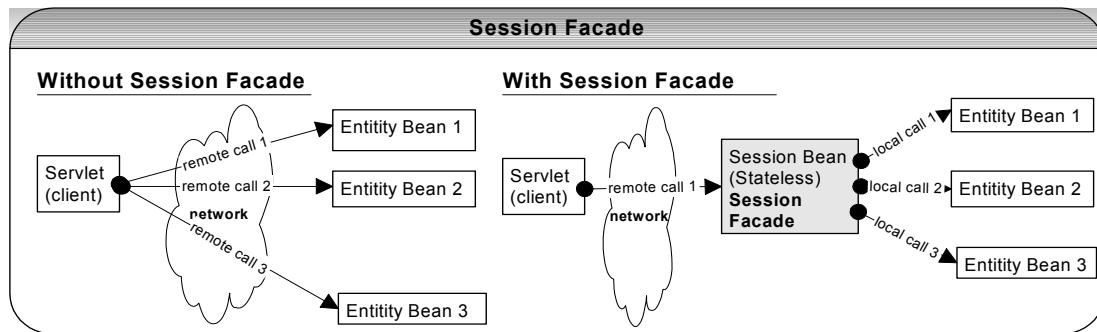
- Invoking session beans in Session Facade.
- Acting as a service locator and cache home stubs to improve performance.
- Handling exceptions from the server side. (Unchecked exceptions get wrapped into the remote exception, checked exceptions can be thrown as an application exception or wrapped in the remote exception. unchecked exceptions do not have to be caught but can be caught and should not be used in the method signature.)
- Re-trying services for the client (For example when using optimistic locking business delegate will retry the method call when there is a concurrent access.).



**Q 84:** What is a session façade? **[DP PI]**

**A 84: Problem:** Too many method invocations between the client and the server will lead to network overhead, tight coupling due to dependencies between the client and the server, misuse of server business methods due to fine grained access etc.

**Solution:** Use a **session bean** as a **façade** to encapsulate the complexities between the client and the server interactions. The Session Facade manages the business objects, and provides a uniform coarse-grained service access layer to clients.



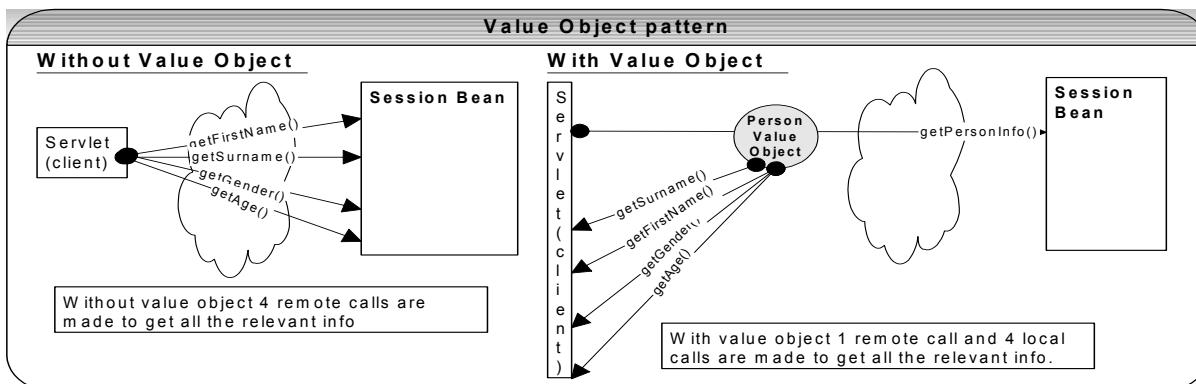
**Session façade** is responsible for

- Improving performance by minimising fine-grained method calls over the network.
- Improving manageability by reducing coupling, exposing uniform interface and exposing fewer methods to clients.
- Managing transaction and security in a centralised manner.

**Q 85:** What is a value object pattern? **[DP PI]**

**A 85: Problem:** When a client makes a remote call to the server, there will be a process of network call and serialization of data involved for the remote invocation. If you make fine grained calls there will be performance degradation.

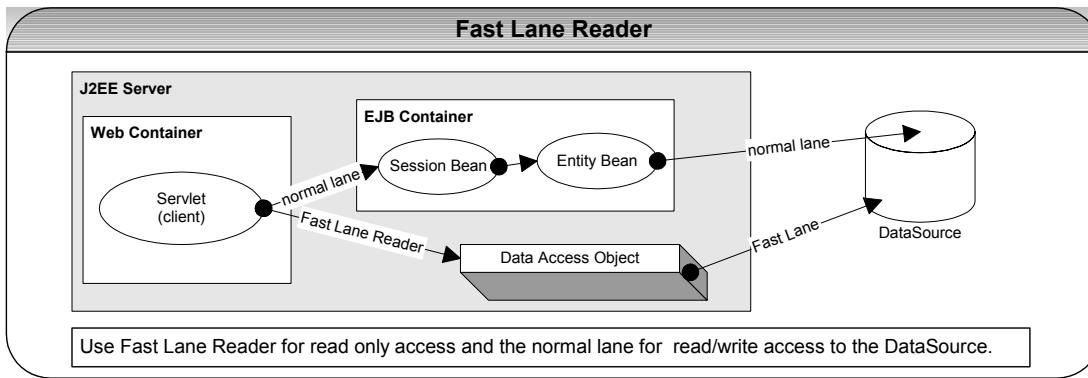
**Solution:** Avoid fine-grained method calls by creating a value object, which will help the client, make a coarse-grained call.



**Q 86:** What is a fast-lane reader? **[DP PI]**

**A 86: Problem:** Using Entity beans to represent persistent, read only tabular data incurs performance cost at no benefit (especially when large amount of data to be read).

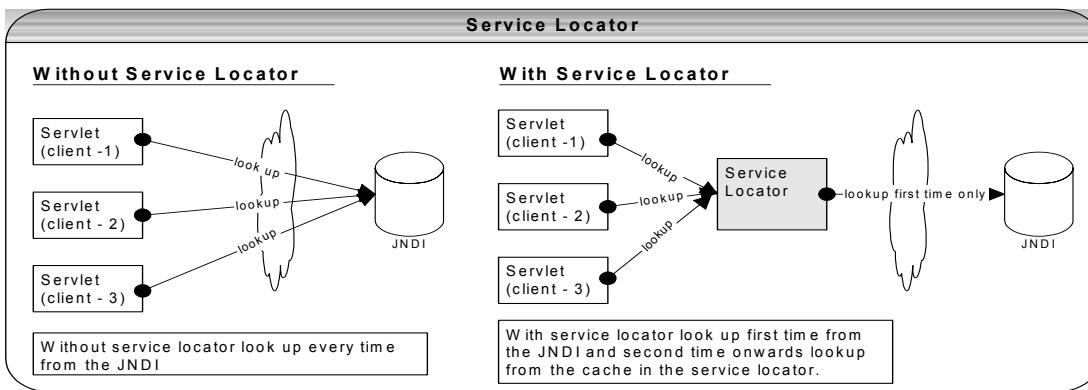
**Solution:** Access the persistent data directly from the database using the DAO (Data Access Object) pattern instead of using Entity beans. The Fast lane readers commonly use JDBC, Connectors etc to access the read-only data from the data source. The main benefit of this pattern is the faster data retrieval.



**Q 87:** What is a Service Locator? **[DP PI]**

**A 87: Problem:** J2EE makes use of the JNDI interface to access different resources like JDBC, JMS, EJB etc. The client looks up for these resources through the JNDI look-up. The JNDI look-up is expensive because the client needs to get a network connection to the server first. So this look-up process is expensive and redundant.

**Solution:** To avoid this expensive and redundant process, service objects can be cached when a client performs the JNDI look-up for the first time and reuse that service object from the cache for the subsequent look-ups. The service locator pattern implements this technique. Refer to diagram below:



**Q 88:** Explain lazy loading and dirty marker strategies? **[DP PI]**

**A 88: Lazy Loading:** Lazy loading means not creating an object until the first time it is accessed. This technique is useful when you have large hierarchies of objects. You can lazy load some of the dependent objects. You only create the dependent (subordinate) objects only when you need them.

```
If (this.data == null) {
 //lazy load data
}
```

For a CMP bean the default scenario is set to no lazy loading and the finder method will execute a single SQL select statement against the database. So, for example, with the `findAllCustomers()` method will retrieve all customer objects with all the CMP fields in each customer object.

If you turn on lazy loading then only the primary keys of the objects within the finder are returned. Only when you access the object, the container uploads the actual object based on the primary key. You may want to turn on the lazy loading feature if the number of objects that you are retrieving is so large that loading them all into local cache would adversely affect the performance. (**Note:** The implementation of lazy loading strategy may vary from container vendor to vendor).

**Dirty Marker (Store optimisation):** This strategy allows us to persist only the entity beans that have been modified. The dependent objects need not be persisted if they have not been modified. This is achieved by using a dirty flag to mark an object whose contents have been modified. The container will check every dependent object and will persist only those objects that are dirty. Once it is persisted its dirty flag will be cleared. (**Note:** The implementation of dirty marker strategy may vary from container vendor to vendor).

**Note:** If your job requires a very good understanding of EJB 2.x then following books are recommended:

- **Mastering Enterprise JavaBeans** – by Ed Roman
- **EJB Design Patterns** – by Floyd Marinescu

## Enterprise - JMS

**Q 89:** What is Message Oriented Middleware? What is JMS? **SF**

**A 89:** Message Oriented Middleware (MOM) is generally defined as a software infrastructure that asynchronously communicates with other disparate systems through the production and consumption of messages. A message may be a request, a report, or an event sent from one part of an enterprise application to another.

Messaging enables loosely coupled distributed communication. A component sends a message to a destination, and the recipient can retrieve the message from the destination. However, the sender and the receiver do not have to be available at the same time in order to communicate and also they are not aware of each other. In fact, the sender does not need to know anything about the receiver; nor does the receiver need to know anything about the sender. The sender and the receiver need to know only what message format and what destination to use. In this respect, messaging differs from tightly coupled technologies, such as Remote Method Invocation (RMI), which requires an application to know a remote application's methods.

Message Oriented Middleware (MOM) systems like MQSeries, MQSonic, etc are proprietary systems. Java Message Service (JMS) is a Java API that allows applications to create, send, receive, and read messages in a standard way. Designed by Sun and several partner companies, the JMS API defines a common set of interfaces and associated semantics that allow programs written in the Java programming language to communicate with other messaging implementations (e.g. MQSonic, TIBCO etc). The JMS API minimises the set of concepts a programmer must learn to use messaging products but provides enough features to support sophisticated messaging applications. It also strives to maximise the portability of JMS applications across JMS providers.

Companies have spent decades developing their legacy systems. Rather than throwing these systems out, XML can be used in a non-proprietary way to move data from legacy systems to distributed systems like J2EE over the wire-using MOM and JMS.

### How JMS is different from RPC?

Remote Procedure Call (e.g. RMI)	JMS
Remote Procedure Call (RPC) technologies like RMI attempt to mimic the behaviour of system that runs in one process. When a remote procedure is invoked the caller is blocked until the procedure completes and returns control to the caller. This is a synchronized model where process is performed sequentially ensuring that tasks are completed in a predefined order. The synchronized nature of RPC tightly couples the client (the software making the call) to the server (the software servicing the call). The client can not proceed (its blocked) until the server responds. The tightly coupled nature of RPC creates highly interdependent systems where a failure on one system has an immediate impact on other systems.	With the use of Message Oriented Middleware (MOM), problems with the availability of subsystems are less of an issue. A fundamental concept of MOM is that communications between components is intended to be asynchronous in nature. Code that is written to connect the pieces together assumes that there is a one-way message that requires no immediate response. In other words, there is no blocking. Once a message is sent the sender can move on to other tasks; it doesn't have to wait for a response. This is the major difference between RPC and asynchronous messaging and is critical to understanding the advantages offered by MOM systems.  In an asynchronous messaging system each subsystem (Customer, Account etc) is decoupled from the other systems. They communicate through the messaging server, so that a failure in one does not impact the operation of the others.

Client is blocked while it is being processed.	Asynchronous messages also allows for parallel processing i.e. client can continue processing while the previous request is being satisfied.
------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------

**Are messaging applications slow?** While there is some overhead in all messaging systems, but this does not mean that the applications that are using messaging are necessarily slow. Messaging systems can achieve a throughput of 100 messages per second depending on the installation, messaging modes (synchronous versus asynchronous, persistent versus non-persistent), and acknowledgment options such as *auto mode*, *duplicates okay mode*, and *client mode* etc. The asynchronous mode can significantly boost performance by multi-tasking. **For example:** In an Internet based shopping cart application, while a customer is adding items to his/her shopping cart, your application can trigger an inventory checking component, and a customer data retrieval component to execute concurrently.

**Are messaging applications reliable?** This is basically a trade-off between performance and reliability. If reliability is more important then the:

- Acknowledgment option should be set to *automode* where once only delivery is guaranteed
- Message delivery mode should be set to *persistent* where the MOM writes the messages to a secure storage like a database or a file system to insure that the message is not lost in transit due to a system failure.

#### What are some of the key message characteristics defined in a message header?

Characteristic	Explanation
JMSCorrelationID	Used in request/response situations where a JMS client can use the JMSCorrelationID header to associate one message with another. <b>For example:</b> a client request can be matched with a response from a server based on the JMSCorrelationID.
JMSSessageID	Uniquely identifies a message in the MOM environment.
JMSDeliveryMode	This header field contains the delivery modes: PERSISTENT or NON_PERSISTENT.
JMSExpiration	This contains the time-to-live value for a message. If it is set to zero, then a message will never expire.
JMSPriority	Sets the message priority but the actual meaning of prioritization is MOM vendor dependent.

**What are the different body types (aka payload types) supported for messages?** All JMS messages are read-only once posted to a queue or a topic.

- **Text message:** body consists of java.lang.String.
- **Map message:** body consists of key-value pairs.
- **Stream message:** body consists of streams of Java primitive values, which are accessed sequentially. XML documents make use of this type.
- **Object message:** body consists of a Serializable Java object.
- **Byte message:** body consists of arbitrary stream of bytes.

#### What is a message broker?

A message broker acts as a server in a MOM. A message broker performs the following operations on a message it receives:

- Processes message header information.
- Performs security checks and encryption/decryption of a received message.
- Handles errors and exceptions.
- Routes message header and the payload (aka message body).
- Invokes a method with the payload contained in the incoming message (e.g. calling onMessage(..) method on a Message Driven Bean (MDB)).
- Transforms the message to some other format. For example XML payload can be converted to other formats like HTML etc with XSLT.

**Q 90:** What type of messaging is provided by JMS? **SF**

**A 90: Point-to-Point:** provides a traditional **queue** based mechanism where the client application sends a message through a queue to typically one receiving client that receives messages sequentially. A JMS message queue is an administered object that represents the message destination for the sender and the message source for the receiver.

**Publish/Subscribe:** is a one-to-many publishing model where client applications publish messages to **topics**, which are in turn subscribed by other interested clients. All subscribed clients will receive each message.

**Q 91:** Discuss some of the design decisions you need to make regarding your message delivery? **SF DC**

**A 91:**

During your design phase, you should carefully consider various options or modes like message acknowledgment modes, transaction modes and delivery modes. **For example:** for a simple approach you would not be using transactions and instead you would be using acknowledgment modes. If you need reliability then the delivery mode should be set to persistent. This can adversely affect performance but reliability is increased.

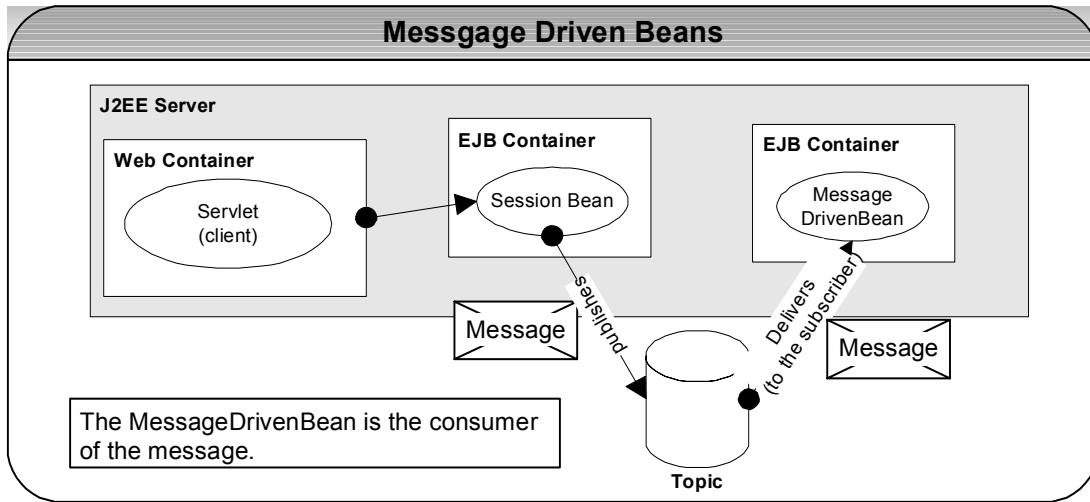
Design decision	Explanation
Message acknowledgment options or modes.	<p><b>Acknowledgement mode and transaction modes</b> are used to determine if a message will be lost or re-delivered on failure during message processing by the target application. Acknowledgment modes are set when creating a JMS session.</p> <pre>InitialContext ic = new InitialContext(...); QueueConnectionFactory qcf = (QueueConnectionFactory)ic.lookup("AccountConnectionFactory"); QueueConnection qc = qcf.createQueueConnection(); QueueSession session = qc.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);</pre> <p>the above code sample, the transaction mode is set to false and acknowledgment mode is set to auto mode. Let us look at acknowledgment modes:</p> <p><b>AUTO_ACKNOWLEDGE:</b> The messages sent or received from the session are automatically acknowledged. This mode also <b>guarantees once only delivery</b>. If a failure occurs while executing <code>onMessage()</code> method of the destination MDB, then the message is re-delivered. A message is automatically acknowledged when it successfully returns from the <code>onMessage(...)</code> method.</p> <p><b>DUPS_OK_ACKNOWLEDGE:</b> This is just like AUTO_ACKNOWLEDGE mode, but under rare circumstances like during failure recovery messages might be delivered more than once. If a failure occurs then the message is re-delivered. This mode has fewer overheads than AUTO_ACKNOWLEDGE mode.</p> <p><b>CLIENT_ACKNOWLEDGE:</b> The messages sent or received from sessions are not automatically acknowledged. The destination application must acknowledge the message receipt. This mode gives an application full control over message acknowledgment at the cost of increased complexity. This can be acknowledged by invoking the <code>acknowledge()</code> method on <code>javax.jms.Message</code> class.</p>
Transaction modes	<p>In JMS, a transaction organizes a message or a group of messages into an atomic processing unit. So, if a message delivery is failed, then the failed message may be re-delivered. Calling the <code>commit()</code> method commits all the messages the session receives and calling the <code>rollback</code> method rejects all the messages.</p> <pre>InitialContext ic = new InitialContext(...); QueueConnectionFactory qcf = (QueueConnectionFactory)ic.lookup("AccountConnectionFactory"); QueueConnection qc = qcf.createQueueConnection(); QueueSession session = qc.createQueueSession(true, -1);</pre> <p>In the above code sample, the transaction mode is set to true and acknowledgment mode is set to -1, which means acknowledgment mode has no use in this mode. Let us look at transaction modes:</p> <p><b>Message Driven Bean (MDB) with container managed transaction demarcation:</b> An MDB participates in a container transaction by specifying the transaction attributes in its deployment descriptor. A transaction automatically starts when the JMS provider removes the message from the destination and delivers it to the MDB's <code>onMessage(...)</code> method. Transaction is committed on successful completion of the <code>onMessage()</code> method. A MDB can notify the container that a transaction should be rolled back by setting the <code>MessageDrivenContext</code> to <code>setRollbackOnly()</code>. When a transaction is rolled back, the message is re-delivered.</p> <pre>public void onMessage(Message aMessage) {     ...     If(someConditionIsTrue) {         mdbContext.setRollbackOnly();     }     else{         //everything is good. Transaction will be committed automatically on completion of onMessage(..)         method.     } }</pre> <p><b>Message Driven Bean (MDB) with bean managed transaction demarcation:</b> A MDB chooses not to</p>

	<p>participate in a container managed transaction and the MDB programmer has to design and code programmatic transactions. This is achieved by creating a <code>UserTransaction</code> object from the MDB's <code>MessageDrivenContext</code> as shown below and then invoking the <code>commit()</code> and <code>rollback()</code> methods on this <code>UserTransaction</code> object.</p> <pre>public void onMessage(Message aMessage) {     UserTransaction uT = mdbContext.getUserTransaction();     uT.begin();     ...     if(someConditionIsTrue) {         uT.rollback();     }     else{         uT.commit();     } }</pre> <p><b>Transacted session:</b> An application completely controls the message delivery by either committing or rolling back the session. An application indicates successful message processing by invoking <code>Session</code> class's <code>commit()</code> method. Also it can reject a message by invoking <code>Session</code> class's <code>rollback()</code> method. This committing or rollback is applicable to all the messages received by the session.</p> <pre>public void process(Message aMessage, QueueSession qs) {     ...     if(someConditionIsTrue) {         qs.rollback();     }     else{         qs.commit();     }     ... }</pre> <p><b>What happens to rolled-back messages?</b></p> <p>Rolled back messages are re-delivered based on the <b>re-delivery count</b> parameter set on the JMS provider. The <b>re-delivery count</b> parameter is very important because some messages can never be successful and this can eventually crash the system. When a message reaches its re-delivery count, the JMS provider can either log the message or forward the message to an error destination. Usually it is not advisable to retry delivering the message soon after it has been rolled-back because the target application might still not be ready. So we can specify a <b>time to re-deliver</b> parameter to delay the re-delivery process by certain amount of time. This time delay allows the JMS provider and the target application to recover to a stable operational condition.</p> <p>Care should be taken <b>not to make use of a single transaction</b> when using the JMS request/response paradigm where a JMS message is sent, followed by the synchronous receipt of a reply to that message. This is because a JMS message is not delivered to its destination until the transaction commits, and the receipt of the reply will never take place within the same transaction.</p> <p><b>Note:</b> when you perform a JNDI lookup for administered objects like connection factories, topics and/or queues, you should use the logical reference <code>java:comp/env/jms</code> as the environment subcontext. It is also vital to release the JMS resources like connection factories, sessions, queues, topics etc when they are no longer required in a <code>try{}</code> and <code>finally{}</code> block.</p>
Message delivery options	<p>What happens, when the messages are with the JMS provider (i.e. MOM) and a catastrophic failure occurs prior to delivering the messages to the destination application? The messages will be lost if they are non-durable. The message's state whether they are lost or not <b>does not depend on acknowledgment modes or transaction modes</b> discussed above. It <b>depends on the delivery mode</b>, which defines whether the message can be durable (aka persistent) or non-durable (aka non-persistent). If you choose the durable delivery mode then the message is stored into a database or a file system by the JMS server before delivering it to the consumer. Durable messages have an adverse effect on performance, but ensure that message delivery is guaranteed.</p> <pre>InitialContext ic = new InitialContext(...); QueueConnectionFactory qcf = (QueueConnectionFactory)ic.lookup("AccountConnectionFactory"); QueueConnection qc = qcf.createQueueConnection(); QueueSession session = qc.createQueueSession(false, Session.AUTO_ACKNOWLEDGE); //senderQueue is an object of type javax.jms.Queue QueueSender sender = session.createSender(senderQueue); Sender.send(message, intDeliveryMode, intPriority, longTimeToLive );</pre>
Best practices	<ul style="list-style-type: none"> <li>▪ A JMS connection represents a TCP/IP connection from the client to the JMS server. A connection is</li> </ul>

to improve performance	<p>a valuable resource, which should be opened at the appropriate time, should be used concurrently by creating and using pool of sessions, and close the connection in a finally{} block when finished.</p> <ul style="list-style-type: none"> <li>▪ Optimize your JMS sessions with the appropriate acknowledgment mode and transaction mode as discussed above and close your sessions when you are finished with them.</li> <li>▪ Choose your message type (text message, byte message, stream message etc) carefully because the size of a message depends on its type and size can affect performance. For example byte messages takes less space than text messages and for object messages you can reduce the serialization cost by marking some of the variables which need not be sent over the network as transient.</li> <li>▪ Optimize your destinations like queues and topics as follows:           <ul style="list-style-type: none"> <li>▪ Choose a non-durable (aka non-persistent) delivery mode where appropriate.</li> <li>▪ Set time to live parameter appropriately after which the message expires.</li> <li>▪ Where applicable receive messages asynchronously (non-blocking call). If you want to receive messages synchronously you can use one of the following methods on the message consumer:</li> </ul> </li> </ul> <p><b>receive():</b> → blocks the call until it receives the next message.  <b>receive(long timeout);</b> → blocks till a timeout occurs.  <b>receiveNoWait();</b> → never blocks.</p>
------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Q 92:** Give an example of a J2EE application using Message Driven Bean with JMS? **SF**

**A 92:**



### Enterprise - XML

**What is XML? And why is XML important?** XML stands for eXtensible Markup Language. XML is a grammatical system for constructing custom markup languages for describing business data, mathematical data, chemical data etc. **XML loosely couples disparate applications or systems utilizing JMS, Web services** etc. XML uses the same building blocks that HTML does: elements, attributes and values. Let's look at why XML is important.

**Scalable:** Since XML is not in a binary format you can create and edit files with anything and it's also easy to debug. XML can be used to efficiently store small amounts of data like configuration files (web.xml, application.xml, struts-config.xml etc) to large company wide data with the help of XML stored in the database.

**Fast Access:** XML documents benefit from their hierarchical structure. Hierarchical structures are generally faster to access because you can drill down to the section you are interested in.

**Easy to identify and use:** XML not only displays the data but also tells you what kind of data you have. The mark up tags identifies and groups the information so that different information can be identified by different application.

**Stylability:** XML is style-free and whenever different styles of output are required the same XML can be used with different style-sheets (XSL) to produce output in XHTML, PDF, TEXT, another XML format etc.

**Linkability, in-line usability, universally accepted standard with free/inexpensive tools etc**

---

**Q 93:** What is the difference between a SAX parser and a DOM parser? **SF PI MI**

**A 93:**

<b>SAX parser</b>	<b>DOM parser</b>
A SAX (Simple API for XML) parser does not create any internal structure. Instead, it takes the occurrences of components of an input document <b>as events</b> (i.e., <b>event driven</b> ), and tells the client what it reads as it reads through the input document.	A DOM (Document Object Model) parser <b>creates a tree structure in memory</b> from an input document and then waits for requests from client.
A SAX parser serves the client application always only with pieces of the document at any given time.	A DOM parser always serves the client application with the entire document no matter how much is actually needed by the client.
A SAX parser, however, is much more space efficient in case of a big input document (because it creates no internal structure). What's more, it runs faster and is easier to learn than DOM parser because its API is really simple. But from the functionality point of view, it provides a fewer functions, which means that the users themselves have to take care of more, such as creating their own data structures.	A DOM parser is rich in functionality. It creates a DOM tree in memory and allows you to access any part of the document repeatedly and allows you to modify the DOM tree. But it is space inefficient when the document is huge, and it takes a little bit longer to learn how to work with it.
Use SAX parser when <ul style="list-style-type: none"> <li>▪ Input document is too big for available memory.</li> <li>▪ When only a part of the document is to be read and we create the data structures of our own.</li> <li>▪ If you use SAX, you are using much less memory and performing much less dynamic memory allocation.</li> </ul>	Use DOM when <ul style="list-style-type: none"> <li>▪ Your application has to access various parts of the document and using your own structure is just as complicated as the DOM tree.</li> <li>▪ Your application has to change the tree very frequently and data has to be stored for a significant amount of time.</li> </ul>
<b>SAX Parser example:</b> Xerces, Crimson etc  Use JAXP (Java API for XML Parsing) which enables applications to parse and transform XML documents independent of the particular XML parser. Code can be developed with one SAX parser in mind and later on can be changed to another SAX parser without changing the application code.	<b>DOM Parser example:</b> XercesDOM, SunDOM, OracleDOM etc.  Use JAXP (Java API for XML Parsing) which enables applications to parse and transform XML documents independent of the particular XML parser. Code can be developed with one DOM parser in mind and later on can be changed to another DOM parser without changing the application code.

**Q 94:** Which is better to store data as elements or as attributes? **SF**

**A 94:** A question arising in the mind of XML/DTD designers is whether to model and encode certain information using an *element*, or alternatively, using an *attribute*. The answer to the above question is not clear-cut. But the general guideline is:

- **Using an element:** <book><title>Lord of the Rings</title>...</book>; If you consider the information in question to be part of the essential material that is being expressed or communicated in the XML, put it in an element
- **Using an attribute:** <book title=" Lord of the Rings "/>; If you consider the information to be peripheral or incidental to the main communication, or purely intended to help applications process the main communication, use attributes.

The principle is **data goes in elements and metadata goes in attributes**. Elements are also useful when they contain special characters like "<", ">", etc which are harder to use in attributes.

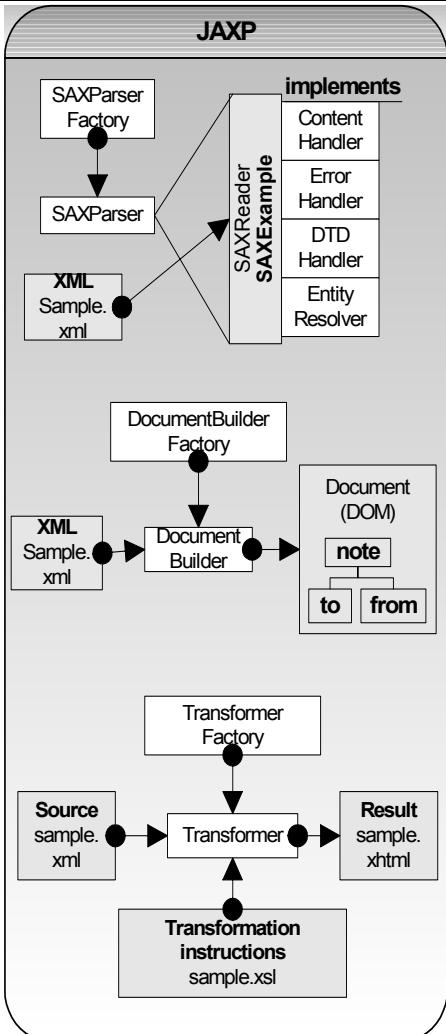
---

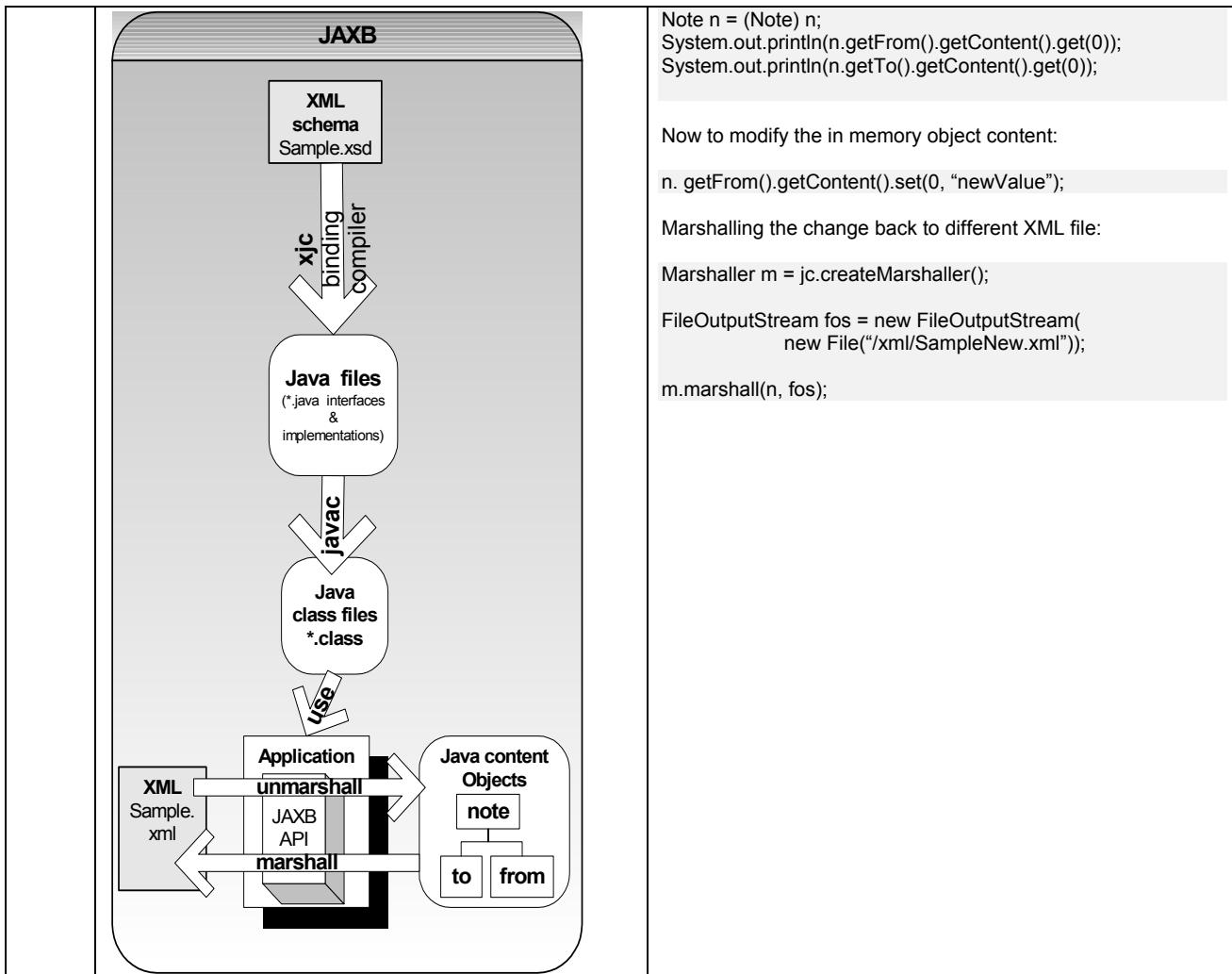
**Q 95:** What is XPATH? What is XSLT/XSL/XSL-FO/XSD/DTD etc? What is JAXB? What is JAXP? **SF**

**A 95:**

What is	Explanation	Example
XML	XML stands for eXtensible Markup Language	<p>Sample.xml</p> <pre>&lt;?xml version="1.0"?&gt; &lt;note&gt;   &lt;to&gt;Peter&lt;/to&gt;   &lt;from&gt;Paul&lt;/from&gt;   &lt;title&gt;Invite&lt;/title&gt;   &lt;content language="English"&gt;Not Much&lt;/content&gt;   &lt;content language="Spanish"&gt;No Mucho&lt;/content&gt; &lt;/note&gt;</pre>
DTD	DTD stands for Document Type Definition. XML provides an application independent way of sharing data. With a DTD, independent groups of people can agree to use a common DTD for interchanging data. Your application can use a standard DTD to verify that data that you receive from the outside world is valid. You can also use a DTD to verify your own data. So the DTD is the building blocks or schema definition of the XML document.	<p>Sample.dtd</p> <pre>&lt;!ELEMENT note (to, from, title, content)&gt; &lt;!ELEMENT to (#PCDATA)&gt; &lt;!ELEMENT from (#PCDATA)&gt; &lt;!ELEMENT title (#PCDATA)&gt; &lt;!ELEMENT content (#PCDATA)&gt; &lt;!ATTLIST content language CDATA #Required&gt;</pre>
XSD	<p>XSD stands for Xml Schema Definition, which is a successor of DTD. So XSD is a building block of an XML document.</p> <p>If you have DTD then why use XSD you may ask?</p> <p>XSD is more powerful and extensible than DTD. XSD has:</p> <ul style="list-style-type: none"> <li>Support for simple and complex data types.</li> <li>Uses XML syntax. So XSD are extensible just like XML because they are written in XML.</li> <li>Better data communication with the help of data types. For example a date like 03-04-2005 will be interpreted in some countries as 3<sup>rd</sup> of April 2005 and in some other countries as 04<sup>th</sup> March 2005.</li> </ul>	<p>Sample.xsd</p> <pre>&lt;?xml version="1.0"?&gt; &lt;xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"   targetNamespace="http://www.w3schools.com"   xmlns="http://www.w3schools.com"   elementFormDefault="qualified"&gt;    &lt;xsd:element name="note"&gt;     &lt;xsd:complexType&gt;       &lt;xsd:sequence&gt;         &lt;xsd:element name="to" type="xsd:string"/&gt;         &lt;xsd:element name="from" type="xsd:string"/&gt;         &lt;xsd:element name="title" type="xsd:string"/&gt;         &lt;xsd:element name="content" type="xsd:string"/&gt;       &lt;/xsd:sequence&gt;     &lt;/xsd:complexType&gt;     &lt;xsd:attribute name="language" type="xsd:string"       use="Required" /&gt;   &lt;/xsd:element&gt; &lt;/xsd:schema&gt;</pre>
XSL	<p><b>XSL</b> stands for eXtensible Stylesheet Language. The XSL consists of 3 parts:</p> <ul style="list-style-type: none"> <li><b>XSLT:</b> Language for transforming XML documents from one to another.</li> <li><b>XPath:</b> Language for defining the parts of an XML document.</li> <li><b>XSL-FO:</b> Language for formatting XML documents. For example to convert an XML document to a PDF document etc.</li> </ul> <p>XSL can be thought of as a set of languages that can :</p> <ul style="list-style-type: none"> <li>Define parts of an XML</li> <li>Transform an XML document to XHTML (eXtensible Hyper Text Markup Language) document.</li> <li>Convert an XML document to a PDF document.</li> <li>Filter and sort XML data.</li> </ul> <p><b>XSLT processor example:</b> Xalan (From Apache)</p> <p><b>PDF Processor example:</b> FOP (Formatting Objects Processor from Apache)</p>	<p>To convert the Sample.xml file to a XHTML file let us apply the following Sample.xsl through XALAN parser.</p> <p>Sample.xsl</p> <pre>&lt;?xml version="1.0"?&gt; &lt;xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl"&gt;   &lt;xsl:template match="/"&gt;     &lt;xsl:apply-templates select="note" /&gt;   &lt;/xsl:template&gt;    &lt;xsl:template match="note"&gt;     &lt;html&gt;       &lt;head&gt;         &lt;title&gt;&lt;xsl:value-of           select="content/@language" /&gt;&lt;/title&gt;       &lt;/head&gt;       &lt;body&gt;         &lt;p&gt;Content:&lt;/p&gt;         &lt;p&gt;&lt;xsl:value-of select="content" /&gt;&lt;/p&gt;       &lt;/body&gt;     &lt;/html&gt;   &lt;/xsl:template&gt; &lt;/xsl:stylesheet&gt;</pre> <p>You get the following output XHTML file:</p> <p>Sample.xhtml</p> <pre>&lt;html&gt;   &lt;head&gt;     &lt;title&gt;English&lt;/title&gt;</pre>

		<pre>&lt;/head&gt; &lt;/html&gt;</pre> <p>Now to convert the Sample.xml into a PDF file apply the following FO (Formatting Objects) file Through the FOP processor.</p> <p>Sample.fo</p> <pre>&lt;?xml version="1.0" encoding="ISO-8859-1"?&gt; &lt;fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"&gt;  &lt;fo:layout-master-set&gt;   &lt;fo:simple-page-master master-name="A4"&gt;     &lt;/fo:simple-page-master&gt;   &lt;/fo:layout-master-set&gt;    &lt;fo:page-sequence master-reference="A4"&gt;     &lt;fo:flow flow-name="xsl-region-body"&gt;       &lt;fo:block&gt;         &lt;xsl:value-of select="content[@language='English']"/&gt;       &lt;/fo:block&gt;     &lt;/fo:flow&gt;   &lt;/fo:page-sequence&gt; &lt;/fo:root&gt;</pre> <p>which gives a basic Sample.pdf which has the following line</p> <p>Not Much</p>
XPath	<p><b>XPath</b> Xml Path Language, a language for addressing parts of an XML document, designed to be used by both XSLT and XPointer. We can write both the patterns (context-free) and expressions using the XPATH Syntax. XPATH is also used in XQuery.</p>	<p>As per Sample.xls</p> <pre>&lt;xsl:template match="content[@language='English']"&gt;   ..... &lt;td&gt;&lt;xsl:value-of select="content/@language" /&gt;&lt;/td&gt;</pre>
JAXP	<p>Stands for Java API for XML Processing. This provides a common interface for creating and using SAX, DOM, and XSLT APIs in Java regardless of which vendor's implementation is actually being used (just like the JDBC, JNDI interfaces). JAXP has the following packages:</p>	<p>DOM example using JAXP:</p> <pre>DocumentBuilderFactory dbf =   DocumentBuilderFactory.newInstance(); DocumentBuilder db = dbf.newDocumentBuilder(); Document doc =   db.parse(new File("xml/Test.xml")); NodeList nl = doc.getElementsByTagName("to"); Node n = nl.item(0); System.out.println(n.getFirstChild().getNodeValue());</pre> <p>SAX example using JAXP:</p> <pre>SAXParserFactory spf =   SAXParserFactory.newInstance(); SAXParser sp = spf.newSAXParser(); SAXExample se = new SAXExample(); sp.parse(new File("xml/Sample.xml"),se);</pre> <p>where SAXExample.Java code snippet</p> <pre>public class SAXExample extends DefaultHandler {    public void startElement(     String uri,     String localName,     String qName,     Attributes attr)     throws SAXException {     System.out.println("&gt;&gt;&gt;" + qName);   }   ... }</pre>

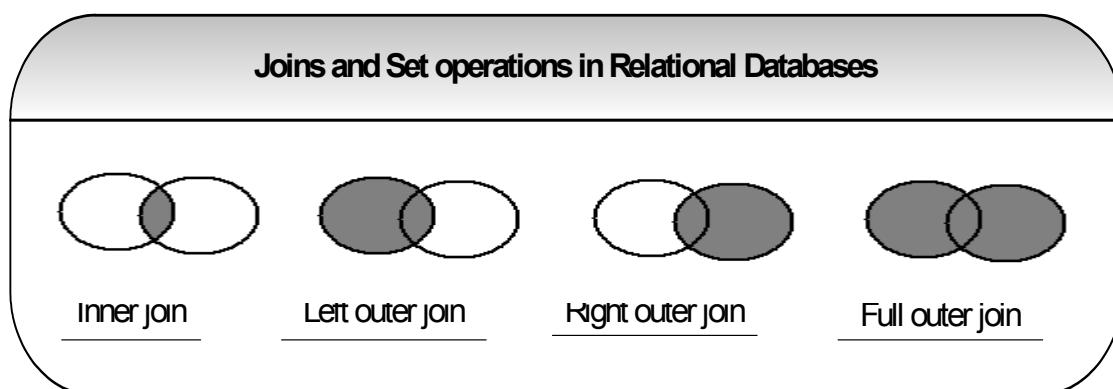
 <p>The diagram illustrates the JAXP architecture with three parallel processing paths:</p> <ul style="list-style-type: none"> <li><b>SAXParser Path:</b> An XML file ("Sample.xml") is processed by a SAXParserFactory to create a SAXParser. This parser then processes the XML file.</li> <li><b>DocumentBuilder Path:</b> An XML file ("Sample.xml") is processed by a DocumentBuilderFactory to create a DocumentBuilder. This builder creates a Document (DOM) object, which contains a "note" element with "to" and "from" children.</li> <li><b>Transformer Path:</b> An XML file ("sample.xml") is processed by a TransformerFactory to create a Transformer. This transformer takes "Transformation instructions" ("sample.xsl") and a "Source" ("sample.xml") to produce a "Result" ("sample.xhtml").</li> </ul>	<p>The DefaultHandler implements ContentHandler, DTDHandler, EntityResolver, ErrorHandler</p> <p>XSLT example using JAXP:</p> <pre>StreamSource xml =     new StreamSource(new File("/xml/Sample.xml")); StreamSource xsl = new StreamSource(     new File("xml/Sample.xsl")); StreamResult result =     new StreamResult(new File("xml/Sample.xhtml"));  TransformerFactory tf =     TransformerFactory.newInstance(); Transformer t = tf.newTransformer(xsl); t.transform(xml, result);</pre> <p>This gives you Sample.xhtml</p> <pre>&lt;html&gt; &lt;head&gt; &lt;title&gt;English&lt;/title&gt; &lt;/head&gt; &lt;/html&gt;</pre> <ul style="list-style-type: none"> <li>Required JAR files are jaxp.jar, dom.jar, xalan.jar, xercesImpl.jar.</li> </ul>
<b>JAXB</b>	<p>Stands for Java API for XML Binding. This standard defines a mechanism for writing out Java objects as XML (marshalling) and for creating Java objects from XML structures (unmarshalling). (You compile a class description to create the Java classes, and use those classes in your application.)</p> <p>Lets look at some code:</p> <p>For binding:</p> <pre>xjc.sh -p com.binding sample.xsd -d work</pre> <p>-p identifies the package for the generated Java files (ie *.Java)</p> <p>-d option identifies the target.</p> <p>Unmarshalling the XML document:</p> <pre>JAXBContext jc = JAXBContext.newInstance(     "com.binding"); Unmarshaller um = jc.createUnmarshaller(); Object o = um.unMarshall(     new File("/xml/"));</pre>



### Enterprise – SQL, Tuning and O/R mapping

**Q 96:** Explain inner and outer joins? **SF**

**A 96:** Joins allow database users to combine data from one table with data from one or more other tables (or views, or synonyms). Tables are joined two at a time making a new table containing all possible combinations of rows from the original two tables. Lets take an example (syntax vary among RDBMS):



**Employees table**

<b>Id</b>	<b>firstname</b>	<b>Surname</b>	<b>state</b>
1001	John	Darcy	NSW
1002	Peter	Smith	NSW
1003	Paul	Gregor	NSW
1004	Sam	Darcy	VIC

**Executives table**

<b>Id</b>	<b>firstname</b>	<b>Surname</b>	<b>state</b>
1001	John	Darcy	NSW
1002	Peter	Smith	NSW
1005	John	Gregor	WA

**Inner joins:** Chooses the join criteria using any column names that happen to match between the two tables. The example below displays only the employees who are executives as well.

```
SELECT emp.firstname, exec.surname FROM employees emp, executives exec
WHERE emp.id = exec.id;
```

*The output is:*

<b>Firstname</b>	<b>surname</b>
John	Darcy
Peter	Smith

**Left Outer joins:** A problem with the inner join is that only rows that match between tables are returned. The example below will show all the employees and fill the null data for the executives.

```
SELECT emp.firstname, exec.surname FROM employees emp left join executives exec
ON emp.id = exec.id;
```

*On oracle*

```
SELECT emp.firstname, exec.surname FROM employees emp, executives exec
where emp.id = exec.id(+);
```

*The output is:*

<b>Firstname</b>	<b>surname</b>
John	Darcy
Peter	Smith
Paul	
Sam	

**Right Outer join:** A problem with the inner join is that only rows that match between tables are returned. The example below will show all the executives and fill the null data for the employees.

```
SELECT emp.firstname, exec.surname FROM employees emp right join executives exec
ON emp.id = exec.id;
```

*On oracle*

```
SELECT emp.firstname, exec.surname FROM employees emp, executives exec
WHERE emp.id(+) = exec.id;
```

*The output is:*

<b>Firstname</b>	<b>surname</b>
John	Darcy
Peter	Smith
	Gregor

**Full outer join:** To cause SQL to create both sides of the join

```
SELECT emp.firstname, exec.surname FROM employees emp full join executives exec
ON emp.id = exec.id;
```

*On oracle*

```
SELECT emp.firstname, exec.surname FROM employees emp, executives exec
WHERE emp.id = exec.id (+)
```

UNION

```
SELECT emp.firstname, exec.surname FROM employees emp, executives exec
WHERE emp.id(+) = exec.id
```

**Note:** Oracle9i introduced the ANSI compliant join syntax. This new join syntax uses the new keywords inner join, left outer join, right outer join, and full outer join, instead of the (+) operator.

*The output is:*

Firstname	surname
John	Darcy
Paul	
Peter	Smith
Sam	
	Gregor

**Self join:** A self-join is a join of a table to itself. If you want to find out all the employees who live in the same city as employees whose first name starts with "Peter", then one way is to use a sub-query as shown below:

```
SELECT emp.firstname, emp.surname FROM employees emp WHERE
city IN (SELECT city FROM employees where firstname like 'Peter')
```

The sub-queries can degrade performance. So alternatively we can use a self-join to achieve the same results.

*On oracle*

```
SELECT emp.firstname, emp.surname FROM employees emp, employees emp2
WHERE emp.state = emp2.state
AND emp2.firstname LIKE 'Peter'
```

*The output is:*

Firstname	Surname
John	Darcy
Peter	Smith
Paul	Gregor

**Q 97:** Explain a sub-query? How does a sub-query impact on performance? **SF PI**

**A 97:** It is possible to embed a SQL statement within another. When this is done on the **WHERE** or the **HAVING** statements, we have a subquery construct. What is subquery useful for? It is used to join tables and there are cases where the only way to correlate two tables is through a subquery.

```
SELECT emp.firstname, emp.surname FROM employees emp WHERE
emp.id NOT IN (SELECT id FROM executives);
```

There are performance problems with sub-queries, which may return NULL values. The above sub-query can be re-written as shown below by invoking a **correlated sub-query**:

```
SELECT emp.firstname, emp.surname FROM employees emp WHERE
emp.id NOT EXISTS (SELECT id FROM executives);
```

The above query can be re-written as an outer join for a faster performance as shown below:

```
SELECT emp.firstname, exec.surname FROM employees emp left join executives exec
on emp.id = exec.id AND exec.id IS NULL;
```

The above execution plan will be faster by eliminating the sub-query.

**Q 98:** What is normalization? When to denormalize? **DC PI**

**A 98:** Normalization is a design technique that is widely used as a guide in designing relational databases. Normalization is essentially a two step process that puts data into tabular form by removing repeating groups and then removes duplicated data from the relational tables (Additional reading recommended).

Redundant data wastes disk space and creates maintenance problems. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations which is time consuming and prone to errors. A change to a customer address is much easier to do if that data is stored only in the Customers table and nowhere else in the database.

Inconsistent dependency is a database design that makes certain assumptions about the location of data. For example, while it is intuitive for a user to look in the Customers table for the address of a particular customer, it may not make sense to look there for the salary of the employee who calls on that customer. The employee's salary is related to, or dependent on, the employee and thus should be moved to the Employees table. Inconsistent dependencies can make data difficult to access because the path to find the data may not be logical, or may be missing or broken.

First normal form	Second Normal Form	Third Normal Form
A database is said to be in First Normal Form when all entities have a unique identifier or key, and when every column in every table contains only a single value and doesn't contain a repeating group or composite field.	A database is in Second Normal Form when it is in First Normal Form plus every non-primary key column in the table must depend on the entire primary key, not just part of it, assuming that the primary key is made up of composite columns.	A database is in Third Normal Form when it is in Second Normal Form and each column that isn't part of the primary key doesn't depend on another column that isn't part of the primary key.

#### When to denormalize? Normalize for accuracy and denormalize for performance.

Typically, transactional databases are highly normalized. This means that redundant data is eliminated and replaced with keys in a one-to-many relationship. Data that is highly normalized is constrained by the primary key/foreign key relationship, and thus has a high degree of *data integrity*. Denormalized data, on the other hand, creates redundancies; this means that it's possible for denormalized data to lose track of some of the relationships between atomic data items. However, since all the data for a query is (usually) stored in a single row in the table, it is *much* faster to retrieve.

**Q 99:** How do you implement one-to-one, one-to-many and many-to-many relationships while designing tables? **SF**

**A 99:** **One-to-One relationship** can be implemented as a single table and rarely as two tables with primary and foreign key relationships.

**One-to-Many relationships** are implemented by splitting the data into two tables with primary key and foreign key relationships.

**Many-to-Many relationships** are implemented using join table with the keys from both the tables forming the composite primary key of the junction table.

**Q 100:** How can you performance tune your database? **PI**

**A 100:**

- Denormalize your tables where appropriate.
- Proper use of index columns: An index based on numeric fields is more efficient than an index based on character columns.
- Reduce the number of columns that make up a composite key.
- Proper partitioning of tablespaces and create a special tablespace for special data types like CLOB, BLOB etc.
- Data access performance can be tuned by using stored procedures to crunch data in the database server to reduce the network overhead and also caching data within your application to reduce the number of accesses.

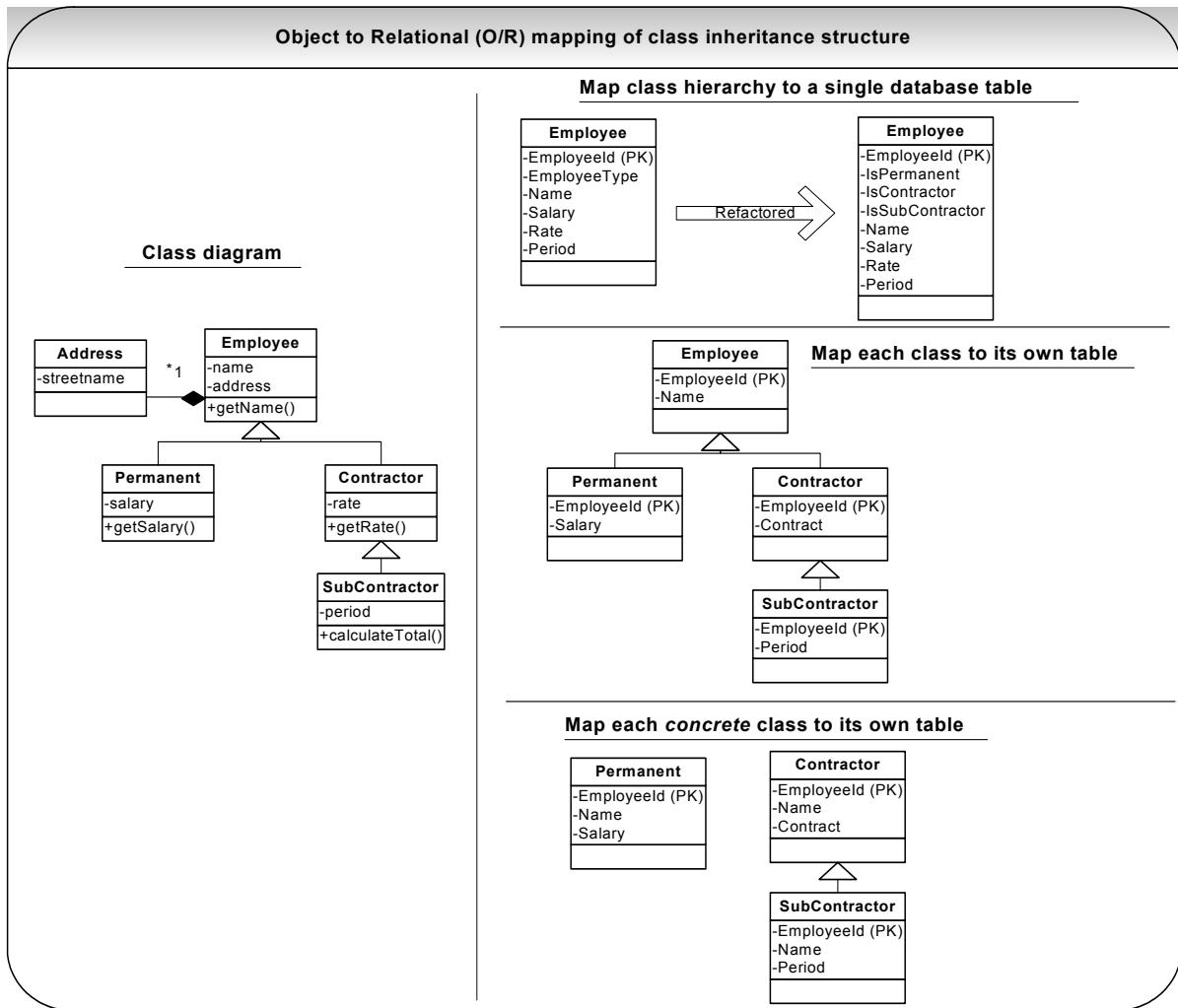
**Q 101:** How will you map objects to a relational database? How will you map class inheritance to relational data model?

**DC**

**A 101:** Due to impedance mismatch between object and relational technology you need to understand the process of mapping classes (objects) and their relationships to tables and relationships between them in a database. Classes represent both behaviour and data whereas relational database tables just implement data. Database schemas have keys (primary keys to uniquely identify rows and foreign keys to maintain relationships between rows) whereas object schema does not have keys and instead use references to implement relationships to other objects. Let us look at some basic points on mapping:

#### Data Modeling :

- Identify entities
- Identify entity attributes
- Apply naming conventions as per logical model standards
- Identify relationships – cardinality
- Apply design patterns
- Assign keys
- Normalize : eliminate data redundancy
- Denormalize : improve performance



- Classes map to tables in a way but not always directly.
- An attribute of a class can be mapped to zero or more columns in a database. Not all attributes are persistent.
- Some attributes of an object are objects itself. For example an *Employee* object has an *Address* object as an attribute. This is basically an **association** relationship between two objects (i.e. *Employee* and *Address*). This is a recursive relationship where at some point the attribute will be mapped to zero or more columns. In this example attributes of the *Address* class will be mapped zero or more columns.
- In its simple form an attribute maps to a single column whereas each has same type (ie attribute is a string and column is a char, or both are dates etc). When you implement mapping with different types (attribute is a currency and column is a float) then you will need to be able to convert them back and forth.

#### How do you map inheritance class structure to relational data model?

Relational databases do not support inheritance. Class inheritance can be mapped to relational tables as follows:

**Map class hierarchy to single database table:** The whole class hierarchy can be stored in a single table by adding an additional column named “EmployeeType”. The column “EmployeeType” will hold the values “Permanent”, “Contract” and “SubContract”. New employee types can be added as required. Although this approach is straightforward it tends to break when you have combinations like an employee is of type both “Contractor” and “SubContractor”. So when you have combinations, you can use refactored table by replacing type code column “EmployeeType” with boolean values such as isPermanent, isContractor and isSubContractor.

**Map each class to its own table:** You create one table per class. The data for a permanent employee is stored in two tables (Employee and Permanent), therefore to retrieve this data you need to join these two tables. To support additional employee type say a *Contractor*, add a new table.

**Map each concrete class to its own table:** You create one table per concrete class. There are tables corresponding to each class like Permanent, Contractor and SubContractor. So join is not required. To support additional employee type, add a new table.

**So which approach to use?** Easiest approach is to have one table per hierarchy and easy to refactor. If you need a “pure design approach” then use one table per class approach. Try to stay away from one table per concrete class approach because it makes refactoring difficult by copying data back and forth between tables. No approach is ideal for all situations.

Another option for mapping inheritance into relational database is to take a generic meta-data driven approach. This approach supports all forms of mapping. In this approach, value of a single attribute will be stored as a row in a table called “Value”. So, to store 5 attributes you need 5 rows in “Value” table. You will have a table called “Class” where class names are stored, a table called “Inheritance” where subclass and superclass information is stored, a table called “Attributes” where class attributes are stored and an “AttributeType” lookup table.

**Q 102:** What is a view? Why will you use a view? What is an aggregate function? Etc. **SF PI**

**A 102:**

Question	Explanation																														
What is view? Why use a view?	<p>View is a precompiled SQL query, which is used to select data from one or more tables. A view is like a table but it doesn't physically take any space (ie not materialised). Views are used for</p> <ul style="list-style-type: none"> <li>▪ Providing inherent security by exposing only the data that is needed to be shown to the end user.</li> <li>▪ Enabling re-use of SQL statements.</li> <li>▪ Allows changes to the underlying tables to be hidden from clients, aiding maintenance of the database schema (i.e. encapsulation).</li> </ul> <p>Views with multiple joins and filters can dramatically degrade performance because views contain no data and any retrieval needs to be processed. The solution for this is to use materialised views or create de-normalised tables to store data. This technique is quite handy in overnight batch processes where a large chunk of data needs to be processed. Normalised data can be read and inserted into some temporary de-normalised table and processed with efficiency.</p>																														
Explain aggregate SQL functions?	<p>SQL provides aggregate functions to assist with the summarisation of large volumes of data.</p> <p>We'll look at functions that allow us to add and average data, count records meeting specific criteria and find the largest and smallest values in a table.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>ORDERID</th><th>FIRSTNAME</th><th>SURNAME</th><th>QTY</th><th>UNITPRICE</th></tr> </thead> <tbody> <tr> <td>1001</td><td>John</td><td>Darcy</td><td>25</td><td>10.5</td></tr> <tr> <td>1002</td><td>Peter</td><td>Smith</td><td>25</td><td>10.5</td></tr> <tr> <td>1003</td><td>Sam</td><td>Gregory</td><td>25</td><td>10.5</td></tr> </tbody> </table> <p><b>SELECT SUM(QTY) AS Total FROM Orders;</b></p> <p><i>The output is:</i> Total = 75</p> <p><b>SELECT AVG(UnitPrice * QTY) As AveragePrice FROM Orders;</b></p> <p><i>The output is:</i> AveragePrice = 262.50</p> <p>If we inserted another row to the above table:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>ORDERID</th><th>FIRSTNAME</th><th>SURNAME</th><th>QTY</th><th>UNITPRICE</th></tr> </thead> <tbody> <tr> <td>1004</td><td>John</td><td>Darcy</td><td>20</td><td>10.50</td></tr> </tbody> </table> <p><b>SELECT FIRSTNAME,SUM(QTY) FROM orders GROUP BY FIRSTNAME HAVING SUM(QTY)&gt;25;</b></p> <p><i>The output is:</i> John 45</p>	ORDERID	FIRSTNAME	SURNAME	QTY	UNITPRICE	1001	John	Darcy	25	10.5	1002	Peter	Smith	25	10.5	1003	Sam	Gregory	25	10.5	ORDERID	FIRSTNAME	SURNAME	QTY	UNITPRICE	1004	John	Darcy	20	10.50
ORDERID	FIRSTNAME	SURNAME	QTY	UNITPRICE																											
1001	John	Darcy	25	10.5																											
1002	Peter	Smith	25	10.5																											
1003	Sam	Gregory	25	10.5																											
ORDERID	FIRSTNAME	SURNAME	QTY	UNITPRICE																											
1004	John	Darcy	20	10.50																											
Explain INSERT, UPDATE, and DELETE statements?	<p><b>INSERT</b> statements can be carried out several ways:</p> <p><b>INSERT INTO ORDERS values (1004, 'John', 'Darcy', 20, 10.50);</b></p>																														

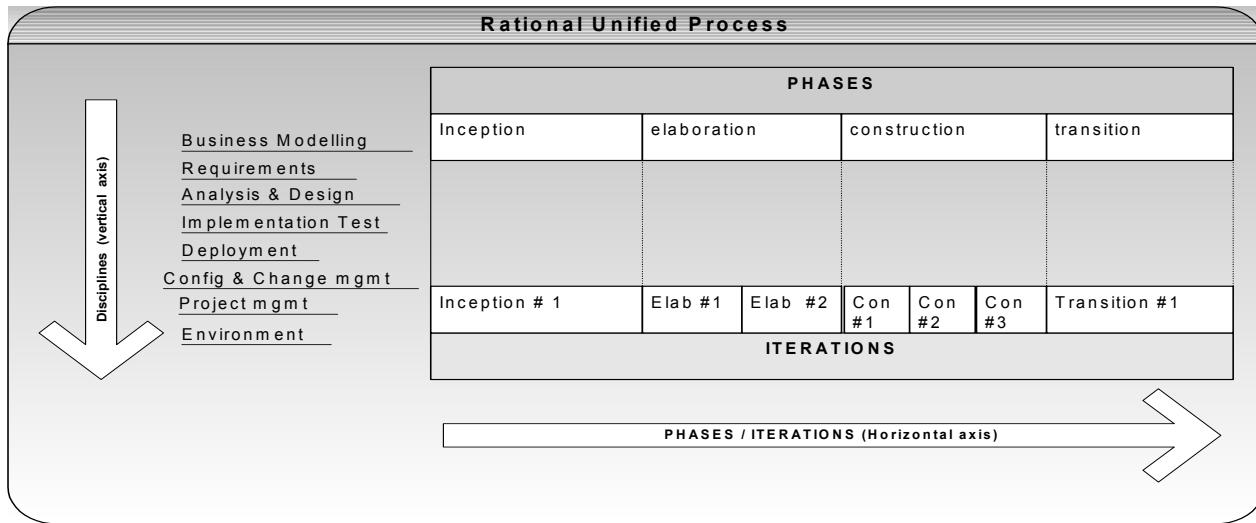
	<p>The above statement is fine but the one below is recommended since it is less ambiguous and less prone to errors.</p> <pre><b>INSERT</b> INTO ORDERS (orderid, firstname, surname, qty, unitprice)     values (1005, 'John', 'Darcy', 20, 10.50);</pre> <p>We can also use INSERT with the SELECT statements as shown below</p> <pre><b>INSERT</b> into NEW_ORDERS (orderid, firstname, surname, qty, unitprice)     SELECT orderid, firstname, surname, qty, unitprice         FROM orders WHERE orderid = 1004;</pre> <p><b>UPDATE</b> statement allows you to update a single or multiple statements.</p> <pre><b>UPDATE</b> ORDERS set firstname='Peter', surname='Piper'     WHERE orderid=1004;</pre> <p>Also can have more complex updates like</p> <pre><b>UPDATE</b> supplier SET supplier_name = ( SELECT customer.name     FROM customers         WHERE customers.customer_id = supplier.supplier_id) WHERE EXISTS     (SELECT customer.name         FROM customers             WHERE customers.customer_id = supplier.supplier_id);</pre> <p><b>DELETE</b> statements allow you to remove records from the database.</p> <pre><b>DELETE</b> FROM ORDERS WHERE orderid=1004;</pre> <p>We can clear the entire table using</p> <pre><b>TRUNCATE TABLE</b> employees;</pre> <p>When running UPDATE/DELETE care should be taken to include WHERE clause otherwise you can inadvertently modify or delete records which you do not intend to UPDATE/DELETE.</p>
How can you compare a part of the name rather than the entire name?	<p>You can use wild card characters like:</p> <ul style="list-style-type: none"> <li>• * ( % in oracle) → Match any number of characters.</li> <li>• ? ( _ in oracle) → Match a single character.</li> </ul> <p><b>To find all the employees who has “au”:</b></p> <pre>SELECT * FROM employees emp     WHERE emp.firstname LIKE '%au%';</pre>
How do you get distinct entries from a table?	<p>The SELECT statement in conjunction with <b>DISTINCT</b> lets you select a set of distinct values from a table in a database.</p> <pre>SELECT <b>DISTINCT</b> empname FROM emptable</pre>
How can you find the total number of records in a table?	<p>Use the <b>COUNT</b> key word:</p> <pre>SELECT <b>COUNT</b>(*) FROM emp WHERE age&gt;25</pre>
What's the difference between a primary key and a unique key?	<p>Both primary key and unique key enforce uniqueness of the column on which they are defined. But by default primary key creates a <b>clustered index</b> on the column, whereas unique creates a <b>non-clustered index</b> by default. Another major difference is that, primary key doesn't allow NULLs, but unique key allows one NULL only.</p>
What are constraints? Explain different types of constraints.	<p>Constraints enable the RDBMS enforce the integrity of the database automatically, without needing you to create triggers, rule or defaults.</p> <p>Types of constraints: <b>NOT NULL, CHECK, UNIQUE, PRIMARY KEY, FOREIGN KEY</b></p>
What is an index? What are the types of indexes? How many clustered indexes can be created on a table? What are the	<p>The books you read have indexes, which help you to go to a specific key word faster. The database indexes are similar.</p> <p>Indexes are of two types. <b>Clustered indexes</b> and non-clustered indexes. When you</p>

advantages and disadvantages of creating separate index on each column of a table?	<p>create a clustered index on a table, all the rows in the table are stored in the order of the clustered index key. So, there can be only one clustered index per table. <b>Non-clustered indexes</b> have their own storage separate from the table data storage. The row located could be the RowID or the clustered index key, depending up on the absence or presence of clustered index on the table.</p> <p>If you create an index on each column of a table, it improves the query performance, as the query optimizer can choose from all the existing indexes to come up with an efficient execution plan. At the same time, data modification operations (such as INSERT, UPDATE, and DELETE) will become slow, as every time data changes in the table, all the indexes need to be updated. Another disadvantage is that, indexes need disk space, the more indexes you have, more disk space is used.</p>
------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Enterprise - RUP & UML

**Q 103:** What is RUP? **SD**

**A 103:** Rational Unified Process (RUP) is a general framework that can be used to describe a development process. The software development cycle has got 4 phases in the following order **Inception**, **Elaboration**, **Construction**, and **Transition**.



The core of the phases is **state-based**, and the state is determined by what fundamental questions you are trying to answer:

- **Inception** - do you and the customer have a shared understanding of the system?
- **Elaboration** - do you have baseline architecture to be able to build the system?
- **Construction** - are you developing a product?
- **Transition** - are you trying to get the customer to take ownership of the system?

RUP is based on a few important philosophies and principles:

- A software project team should **plan ahead**.
- It should know **where it is going**.
- It should capture project knowledge in a **storable** and **extensible** form.

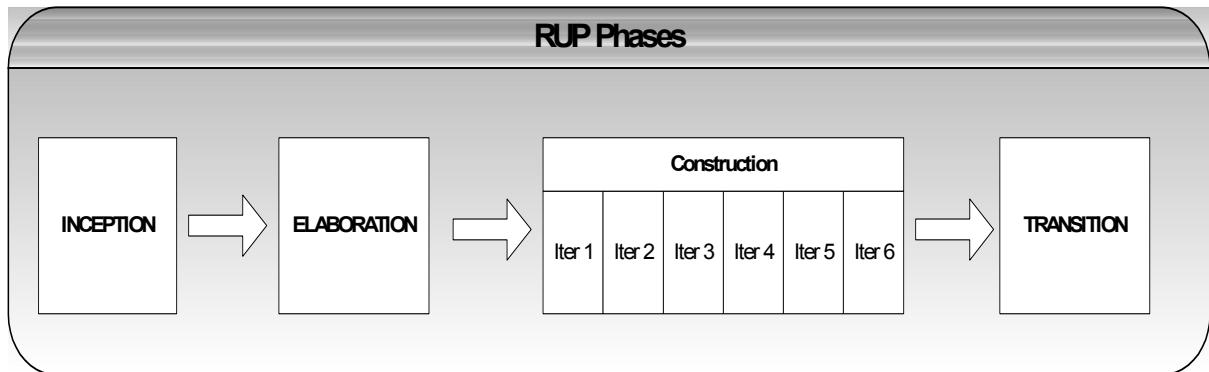
The best practices of RUP involve the following major 5 properties:

Best practice property	Description
Use case driven	Interaction between the users and the system.
Architecture centric	Based on architecture with clear relationships between architectural components.
Iterative	The problem and the solution are divided into more manageable smaller pieces, where each

	iteration will be addressing one of those pieces.
Incremental	Each iteration builds incrementally on the foundation built in the previous iteration.
Controlled	Control with respect to process means you always know what to do next; control with respect to management means that all deliverables, artifacts, and code are under configuration management.

**Q 104:** Explain the 4 phases of RUP? **SD**

**A 104:**



- **Inception:** During the inception phase, you work out the business case for the project. You also will be making a rough cost estimate and return on investment. You should also outline the scope and size of the project.

**The fundamental question you ask at the end of this phase:** do you and the customer have a shared understanding of the system?

- **Elaboration:** At this stage you have the go ahead of the project however only have vague requirements. So at this stage you need to get a better understanding of the problem. Some of the steps involved are:

- What is it you are actually going to build?
- How are you going to build it?
- What technology are you going to use?
- Analysing and dealing with requirement risks, technological risks, skill risks, political risks etc.
- Develop a **domain model**, **use case model** and a **design model**. The UML techniques can be used for the model diagrams (e.g. class diagrams, sequence diagrams etc).

An important result of the elaboration phase is that you have a **baseline architecture**. This architecture consists of:

- A list of use cases depicting the requirements.
- The domain model, which captures your understanding of the domain with the help of UML class diagrams.
- Selection of key implementation technology and how they fit together. For example: Java/J2EE with JSP, Struts, EJB, XML, etc.

**The fundamental question you ask at the end of this phase:** do you have a baseline architecture to be able to build the system?

- **Construction:** In this phase you will be building the system in a series of iterations. Each iteration is a mini project. You will be performing analysis, design, unit testing, coding, system testing, and integration testing for the use cases assigned to each iteration. The iterations within the construction phase are incremental and iterative. Each iteration builds on the use cases developed in the previous iterations. The each iteration will involve code rewrite, refactoring, use of design patterns etc.

The basic documentation required during the construction phase is:

- A **class diagram** and a **sequence diagram**.
- Some text to pull the diagrams together.

- If a class has complex life cycle behaviour then a **state diagram** is required.
- If a class has a complex computation then an **activity diagram** is required.

**The fundamental question you ask at the end of this phase:** do you have a developed product?

- **Transition:** During this phase you will be delivering the finished code regularly. During this phase there is no coding to add functionality unless it is small and essential. There will be bug fixes, code optimisation etc during this phase. An example of a transition phase is that the time between the beta release and the final release of a product.

**The fundamental question you ask at the end of this phase:** are you trying to get the customer to take ownership of the developed product or system?

---

**Q 105:** What are the characteristics of RUP? Where can you use RUP? **SD**

**A 105:**

1. RUP is based on a few important philosophies and principles like planning ahead, knowing where the process is heading and capturing the project in storables and extensible manner.
2. It is largely based on OO analysis and design, and use case driven etc.
3. Iterative and incremental development as opposed to waterfall approach, which hides problems.
4. Architecture centric approach.

RUP is more suited for larger teams of 50-100 people. RUP can also be used as an agile (i.e. lightweight) process for smaller teams of 20-30 people, or as a heavy weight process for larger teams of 50-100 people. Extreme Programming (XP) can be considered as a subset of RUP. At the time of writing, the **agile (i.e lightweight) software development process** is gaining popularity and momentum across organizations. Several methodologies fit under this agile development methodology banner. All these methodologies share many characteristics like **iterative** and **incremental development**, **test driven development**, **stand up meetings to improve communication**, **automatic testing**, **build** and **continuous integration of code** etc. Refer **Q136** in Enterprise Java section.

---

**Q 106:** Why is UML important? **SD DC**

**A 106:** The more complicated the underlying system, the more critical the communication among everyone involved in developing and deploying the software. UML is a software blueprint language for analysts, designers and developers. UML provides a common vocabulary for the business analysts, architects, developers etc.

UML is applicable to the Object Oriented problem solving. UML begins with a **model**; A **model** is an abstraction of the underlying problem. The **domain** is the actual world from which the problem comes. The model consists of **objects**. The objects interact with each other by sending and receiving **messages**. The objects are characterised by **attributes** and **operations** (behaviours). The values of an object's attributes determine its **state**. The **classes** are the blueprints (or like templates) for objects. A class wraps **attributes** and **methods** into a single distinct entity. The objects are the **instances** of classes.

---

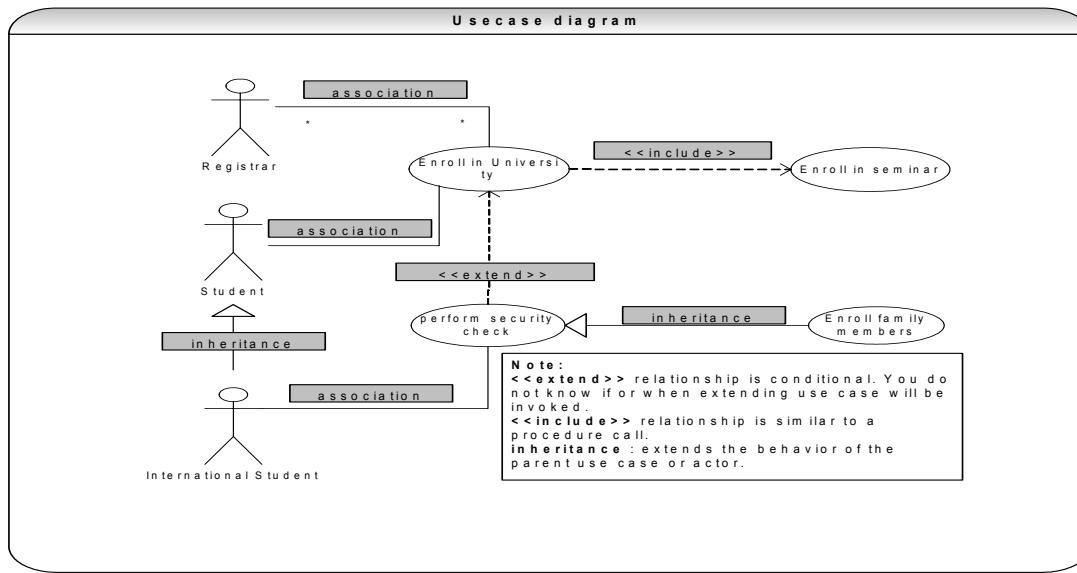
**Q 107:** What are the different types of UML diagrams? **SD DC**

**A 107:** **Use case diagrams:** Depicts the typical interaction between external users (actors) and the system. The emphasis is on **what** a system does rather than **how** it does it. A use case is a summary of scenarios for a single task or goal. An actor is responsible for initiating a task. The connection between actor and use case is a **communication association**.

Capturing use cases is one of the primary tasks of the **elaboration phase** of RUP. In its simplest usage, you capture a use case by talking to your users and discussing the various things they might want to do with the system.

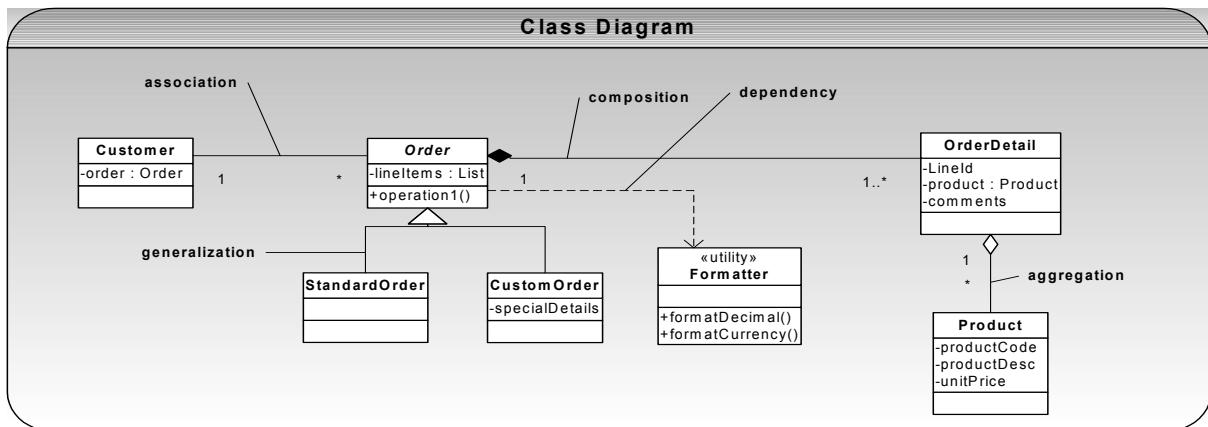
#### When to use 'use case' diagrams?

- Determining user requirements. New use cases often generate new requirements.
- Communicating with clients. The simplicity of the diagram makes use case diagrams a good way for designers and developers to communicate with clients.
- Generating test cases. Each scenario for the use case may suggest a suite of test cases.



**Class diagrams:** Class diagram technique is vital within Object Oriented methods. Class diagrams describe the types of objects in the system and the various static relationships among them. Class diagrams also show the attributes and the methods. Class diagrams have the following possible relationships:

- **Association:** A relationship between instances of 2 classes.
- **Aggregation:** An **association** in which one class belongs to a collection (does not always have to be a collection. You can also have cardinality of "1"). This is a **part of a whole** relationship where the **part** can exist without the **whole**. **For example:** A line item is whole and the products are the parts. If a line item is deleted then the products **need not be deleted**.
- **Composition:** An **association** in which one class belongs to a collection (does not always have to be a collection. You can also have cardinality of "1"). This is a **part of a whole** relationship where the **part cannot** exist without the **whole**. If the whole is deleted then the parts are deleted. **For example:** An Order is a whole and the line items are the parts. If an order is deleted then all the line items **should be deleted** as well (ie cascade deletes).
- **Generalization:** An inheritance link indicating that one class is a superclass of the other. The Generalization expresses the "**is a**" relationship whereas the association, aggregation and composition express the "**has a**" relationship.
- **Dependency:** A dependency is a weak relationship where one class requires another class. The dependency expresses the "**uses**" relationship. **For example:** A domain model class uses a utility class like Formatter etc.

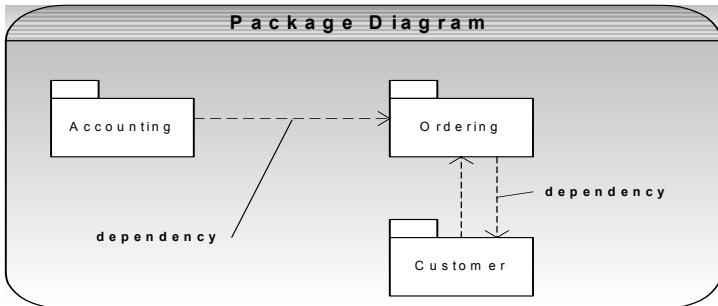


### When to use class diagrams?

- Class diagrams are the backbone of **Object Oriented** methods. So they are used frequently.

- Class diagrams can have a conceptual perspective and an implementation perspective. During the analysis draw the conceptual model and during implementation draw the implementation model.

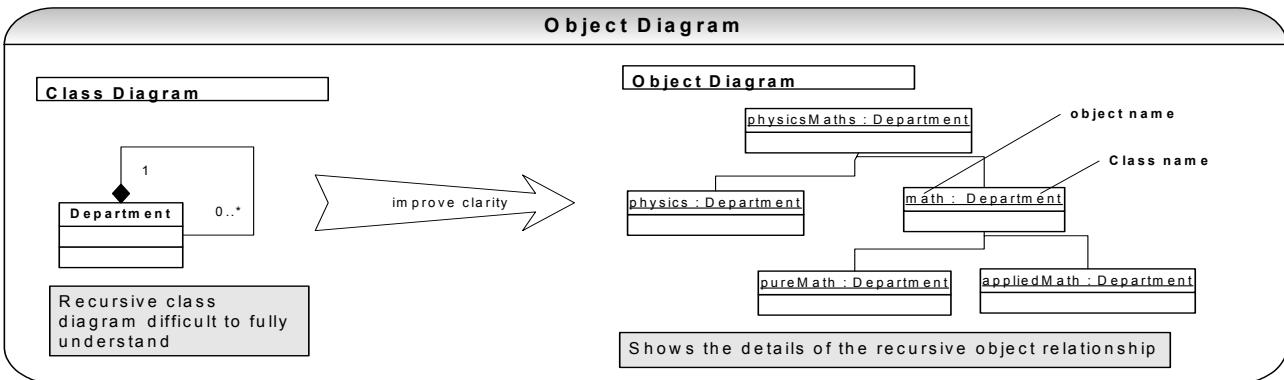
**Package diagrams:** To simplify complex class diagrams you can group classes into **packages**.



### When to use package diagrams?

- Package diagrams are vital for large projects.

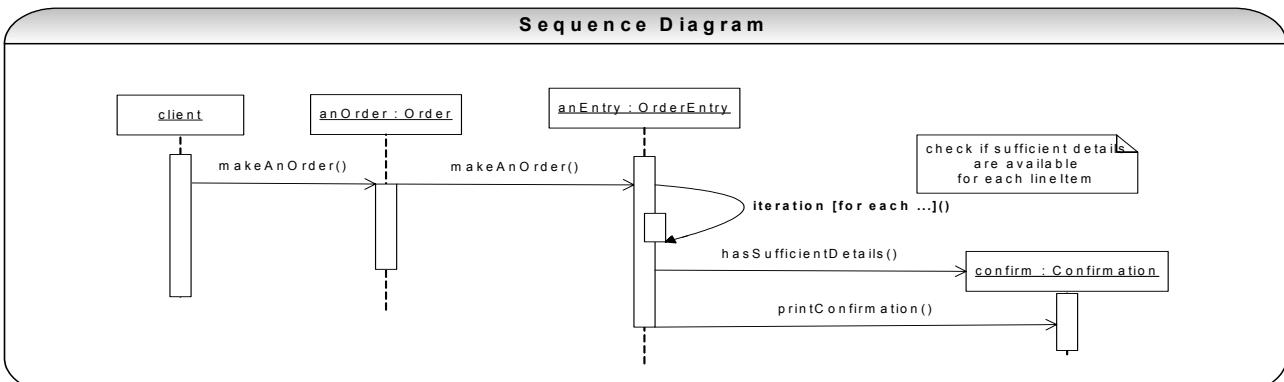
**Object diagrams:** Object diagrams show instances instead of classes. They are useful for explaining some complicated objects in detail about their recursive relationships etc.



### When to use object diagrams?

- Object diagrams are vital for large projects.
- They are useful for explaining structural relationships in detail for complex objects.

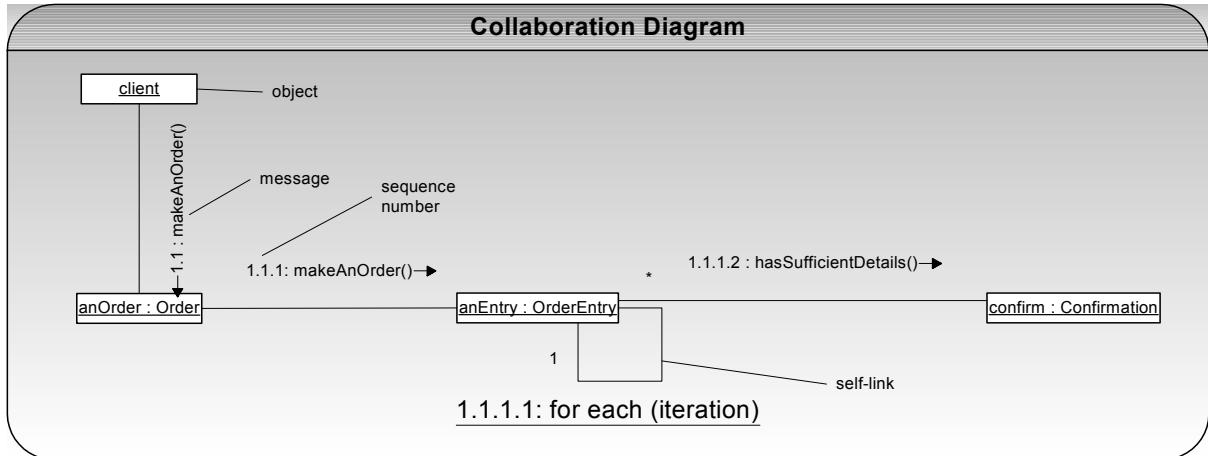
**Sequence diagrams:** Sequence diagrams are interaction diagrams which detail **what** messages are sent and **when**. The sequence diagrams are organized according to time. The time progresses as you move from top to bottom of the diagram. The objects involved in the diagram are shown from left to right according to **when** they take part.



Note: Each vertical dotted line is a **life line**. Each arrow is a **message**. The rectangular boxes on the life line are called the **activation bar** which represents the duration of execution of message.

**Collaboration diagrams:** Collaboration diagrams are also **interaction diagrams**. Collaboration diagrams convey the same message as the sequence diagrams. But the collaboration diagrams focus on the **object roles** instead of the **times** at which the messages are sent.

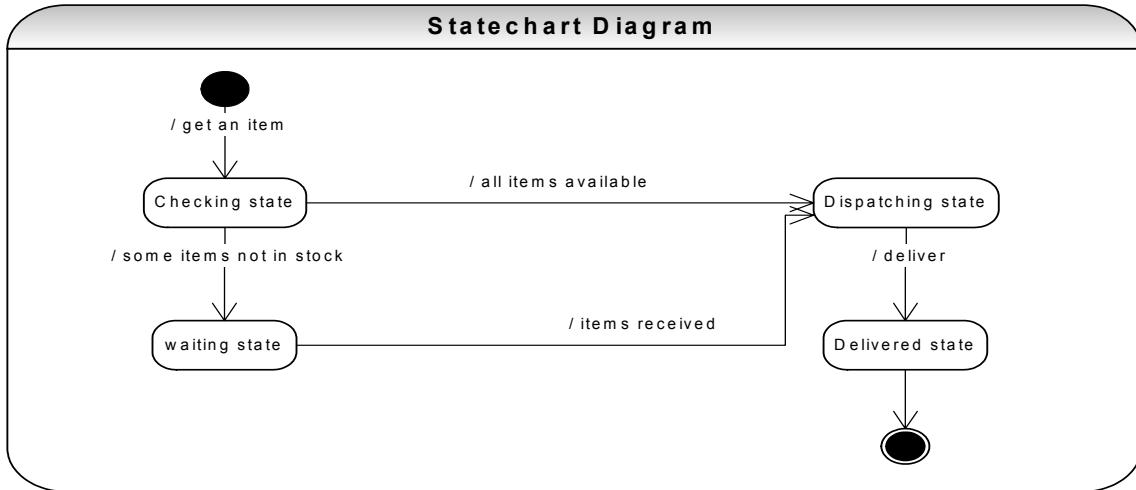
The collaboration diagrams use the decimal sequence numbers as shown in the diagram below to make it clear which operation is calling which other operation, although it can be harder to see the overall sequence. The top-level message is numbered 1. The messages at the same level have the same decimal prefix but different suffixes of 1, 2 etc according to when they occur.



### When to use interaction diagrams?

- When you want to look at behaviour of several objects within a single use case. If you want to look at a single object across multiple use cases then use statechart diagram as described below.

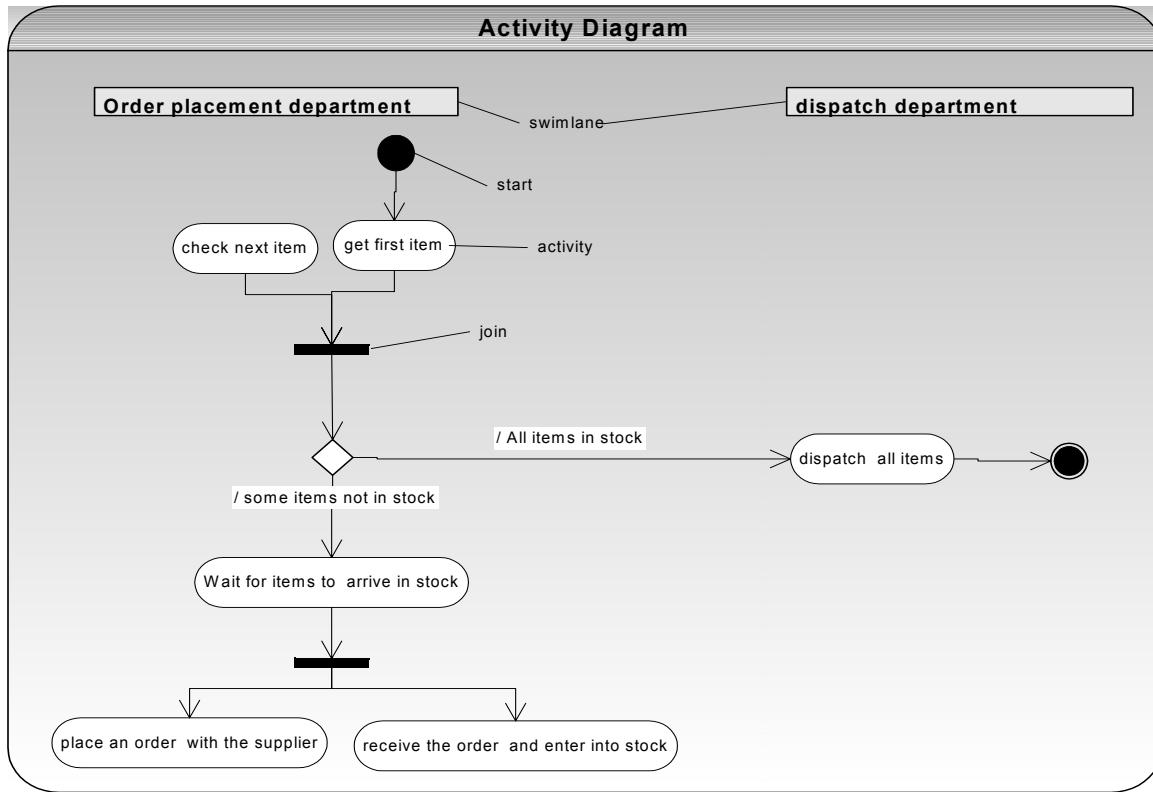
**State chart diagrams:** Objects have behaviour and state. The state of an object depends on its current activity or condition. This diagram shows the possible states of the object and the transitions that cause a change in its state.



### When to use statechart diagram?

- Statechart diagrams are good at describing the behaviour of an object across several use cases. But they are not good at describing the interaction or collaboration between many objects. Use interaction and/or activity diagrams in conjunction with a statechart diagram.
- Use it only for classes that have complex state changes and behaviour. **For example:** the User Interface (UI) control objects, Objects shared by multi-threaded programs etc.

**Activity diagram:** This is really a fancy flow chart. The activity diagram and statechart diagrams are related in a sense that statechart diagram focuses on object undergoing a transition process and an activity diagram focuses on the flow of activities involved in a single transition process.

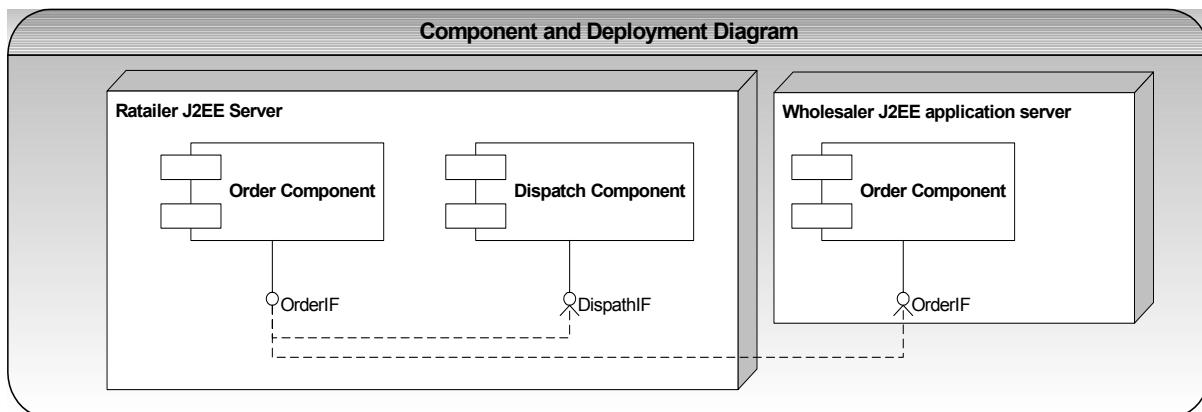


In domain modelling it is imperative that the diagram conveys which object (or class) is responsible for each activity. Activity diagrams can be divided into object **swimlanes** that determine which object is responsible for which activity. The swimlanes are quite useful because they combine the activity diagram's depiction of logic with the interaction diagram's depiction of responsibility. A single transition comes out of each **activity**, connecting to the next activity. A transition may **join** or **fork**.

### When to use activity diagrams?

The **activity** and **statechart** diagrams are generally useful to express complex operations. The great strength of activity diagrams is that they support and encourage parallel behaviour. The activity and statechart diagrams are beneficial for workflow modelling with multi-threaded programming.

**Component and Deployment diagrams:** A component is a code module. Component diagrams are physical diagrams analogous to a class diagram. The deployment diagrams show the physical configuration of software and hardware components. The physical hardware is made up of **nodes**. Each **component** belongs to a node.



**Q 108:** What is the difference between aggregation and composition? **SD DC**

**A 108:**

Aggregation	Composition
<b>Aggregation:</b> An <b>association</b> in which one class belongs to another class or a collection. This is a <b>part of a whole</b> relationship where the <b>part</b> can exist without the <b>whole</b> . <b>For example:</b> A line item is whole and the products are the parts. If a line item is deleted then the products <u>need not be deleted</u> . (no cascade delete in database terms)	<b>Composition:</b> An <b>association</b> in which one class belongs to another class or a collection. This is a <b>part of a whole</b> relationship where the <b>part</b> cannot exist without the <b>whole</b> . If the <b>whole</b> is deleted then the parts are deleted. <b>For example:</b> An Order is a whole and the line items are the parts. If an order is deleted then all the line items <u>should be deleted</u> as well (i.e. cascade deletes in database terms).
Aggregations are not allowed to be circular.	In a garbage-collected language like Java, The <b>whole</b> has the responsibility of <b>preventing</b> the garbage collector to prematurely collect the <b>part</b> by holding reference to it.

**Q 109:** What is the difference between a collaboration diagram and a sequence diagram? **SD DC**

**A 109:** You can automatically generate one from the other.

Sequence Diagram	Collaboration Diagram
The emphasis is on the <b>sequence</b> .	The emphasis is on the <b>object roles</b>

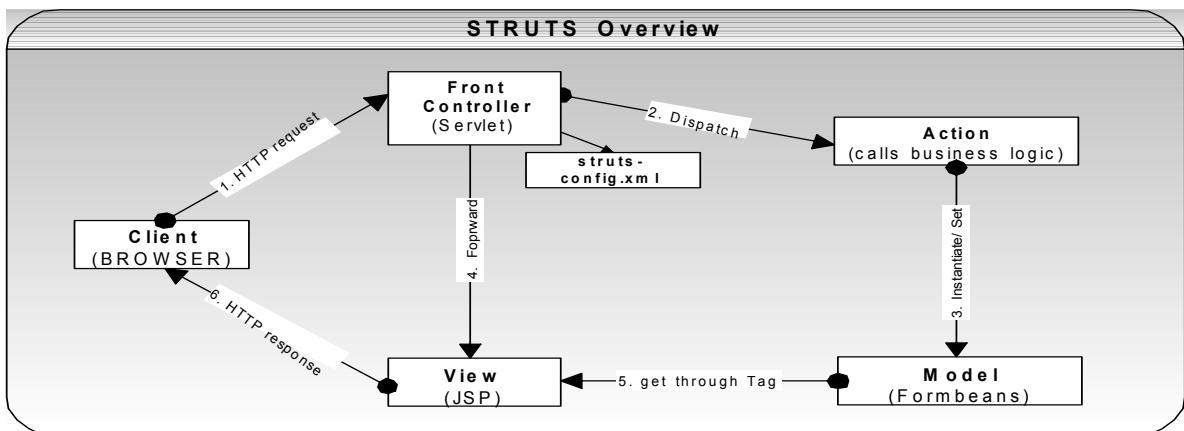
**Reference:** The above section on RUP & UML is based on the book **UML Distilled** by Martin Fowler and Kendall Scott. If you would like to have a good understanding of UML & RUP, then this book is recommended.

## Enterprise - Struts

**Struts** is a Web-based user interface framework, which has been around for a few years. It is a matured and proven framework, which has been used in many J2EE projects. While Struts has been demonstrating its popularity, there is an emerging framework called **JavaServer Faces (JSF)** gaining lots of momentum and popularity. Like Struts, JSF provides Web application life cycle management through a controller servlet, and like Swing, JSF provides a rich component model complete with event handling and component rendering. So JSF can be considered as a combination of Struts frame work and Java Swing user interface framework. Refer **Q19 – Q20** in Emerging Technologies/Frameworks section for JSF.

**Q 110:** Give an overview of Struts? **SF DP**

**A 110:** Struts is a framework with set of cooperating classes, servlets and JSP tags that make up a reusable MVC 2 design.



- **Client (Browser):** A request from the client browser creates an HTTP request. The Web container will respond to the request with an HTTP response, which gets displayed on the browser.

## STRUCTURAL Design Patterns :

**Adapter:** Convert existing interface to new interface in order to achieve reusability of unrelated classes. E.g. wrapper classes

**Composite:** Build complex objects out of elemental objects. E.g. file directory structure

**Decorator:** Add additional functions to existing interface dynamically. E.g. JScrollPane to JTextArea

**Façade:** Make complex system look simple by providing a generic interface

**Proxy:** Simple object to represent complex object, providing a placeholder for the complex object to access it. E.g. stub-skeleton

- **Controller (ActionServlet class and Request Processor class):** The controller receives the request from the browser, and makes the decision where to send the request based on the struts-config.xml. **Design pattern:** Struts controller uses the **command design pattern** by calling the Action classes based on the configuration file struts-config.xml and the RequestProcessor class's process() method uses **template method design pattern** (Refer Q11 in How would you go about ... section) by calling a sequence of methods like:

- **processPath(request, response)** → read the request URI to determine path element.
- **processMapping(request, response)** → use the path information to get the action mapping
- **processRoles(request, response, mapping)** → Struts Web application security which provides an authorization scheme. By default calls request.isUserInRole(). For example allow /addCustomer action if the role is executive.

```
<action path="/addCustomer" roles="executive">
```

- **processValidate(request, response, form, mapping)** → calls the validate() method of the ActionForm.
- **processActionCreate(request, response, mapping)** → gets the name of the action class from the "type" attribute of the <action> element.
- **processActionPerform(req, res, action, form, mapping)** → This method calls the execute method of the Action class which is where business logic is written.
- **Business Logic (Action class):** The Servlet dispatches the request to Action classes, which act as a **thin wrapper to the business logic** (The actual business logic is carried out by either EJB session beans and/or plain Java classes). The action class helps control the **workflow** of the application. (**Note:** The Action class should only control the workflow and not the business logic of the application). The Action class uses the **Adapter design pattern** (Refer Q11 in How would you go about ... section).
- **ActionForm class:** Java representation of HTTP input data. They can carry data over from one request to another, but actually represent the data submitted with the request.
- **View (JSP):** The view is a JSP file. There is no business or flow logic and no state information. The JSP should just have tags to represent the data on the browser.

**ActionServlet** class is the controller part of the MVC implementation and is the core of the framework. It processes user requests, determines what the user is trying to achieve according to the request, pulls data from the model (if necessary) to be given to the appropriate view, and selects the proper view to respond to the user. As discussed above ActionServlet class delegates the grunt of the work to the **RequestProcessor** and **Action** classes.

The **ActionForm** class maintains the state for the Web application. ActionForm is an abstract class, which is subclassed for every input form model. The struts-config.xml file controls, which HTML form request maps to which ActionForm.

The **Action** class is a wrapper around the business logic. The purpose of the Action class is to translate the HttpServletRequest to the business logic. To use the Action class, subclass and overwrite the execute() method. The actual business logic should be in a separate package or EJB to allow reuse of business logic in protocol independent manner (ie the business logic should be used **not only** by HTTP clients **but also** by WAP clients, EJB clients, Applet clients etc).

The **ExceptionHandler** can be defined to execute when the Action class's execute() method throws an Exception. For example

```
<global-exceptions>
 <exception key="my.key" type="java.io.IOException" handler="my.ExceptionHandler"/>
</global-exceptions>
```

When an IOException is thrown then it will be handled by the execute() method of the my.ExceptionHandler class.

The struts-config.xml configuration information is translated into **ActionMapping**, which are put into the **ActionMappings** collection.

Further reading is recommended for more detailed understanding.

**Q 111:** What is a synchronizer token pattern in Struts or how will you protect your Web against multiple submissions?

**DC** **DP**

**A 111:** Web designers often face the situation where a form submission must be protected against duplicate or **multiple submissions**. This situation typically occurs when the user clicks on submit button more than once before the response is sent back or client access a page by returning to the previously bookmarked page.

- The simplest solution that some sites use is that displaying a warning message "Wait for a response after submitting and do not submit twice."
- In the client only strategy, a flag is set on the first submission and from then onwards the submit button is disabled based on this flag. Useful in some situations but this strategy is coupled to the browser type and version etc.
- For a server-based solution the J2EE pattern **synchroniser token pattern** can be applied. The basic idea is to:
  1. Set a token in a session variable on the server side before sending the transactional page back to the client.
  2. The token is set on the page as a hidden field. On submission of the page first check for the presence of a valid token by comparing the request parameter in the hidden field to the token stored in the session. If the token is valid continue processing otherwise take other alternative action. After testing the token must be reset to null.

The synchroniser token pattern is implemented in Struts. How do we implement the alternate course of action when the second click on submit button will cancel the response from the first click. The thread for the first click still runs but has no means of sending the response back to the browser. This means the transaction might have gone through without notifying the user. The user might get the impression that transaction has not gone through.

Struts support for synchronisation comes in the form of:

**ActionServlet.saveToken(HttpServletRequest)** and **ActionServlet.isTokenValid(HttpServletRequest)** etc

**Q 112:** How do you upload a file in Struts? **SF**

**A 112:** In JSP page set the code as shown below: **CO**

```
<html:form action="upload.do" enctype="multipart/form-data" name="fileForm" type="FileForm"
 scope="session">
 Please select the file that you would like to upload:
 <html:file property="file" />
 <html:submit />
</html:form>
```

In the **FormBean** set the code as shown below:

```
public class FileForm extends ActionForm {
 private FormFile file;

 public void setFile(FormFile file){
 this.file = file;
 }

 public FormFile getFile(){
 return file;
 }
}
```

**Q 113:** Are Struts action classes thread-safe? **SF** **CI**

**A 113:** No. Struts action classes are not thread-safe. Struts action classes are cached and reused for performance optimization at the cost of having to implement the action classes in a thread-safe manner.

**Q 114:** How do you implement internationalization in Struts? **SF**

**A 114:** Internationalization is built into Struts framework. In the JSP page set the code as shown below: **CO**

```
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>

<html:html locale="true">
<head>
<title>i18n</title>
</head>

<body>
<h2><bean:message key="page.title"/></h2>
</body>
</html:html>
```

Now we need to create an application resource file named **ApplicationResource.properties**.

page.title=Thank you for visiting!

Now in Italian, create an application resource file named **ApplicationResource\_it.properties**.

page.title=Grazie per la vostra visita!

Finally, add reference to the appropriate resource file in the **struts-config.xml**.

**Q 115:** What is an action mapping in Struts? How will you extend Struts? **SF**

**A 115:** An **action mapping** is a configuration file (struts-config.xml) entry that, in general, associates an action name with an action. An action mapping can contain a reference to a **form bean** that the action can use, and can additionally define a list of local **fowards** that is visible only to this action.

#### How will you extend Struts?

Struts is not only a powerful framework but also very extensible. You can extend Struts in one or more of the following ways:

**Plugin:** Define your own Plugin class if you want to execute some init() and destroy() methods during the application startup and shutdown respectively. Some services like loading configuration files, initialising applications like logging, auditing, etc can be carried out in the init() method.

**RequestProcessor:** You can create your own RequestProcessor by extending the Struts RequestProcessor. For example you can override the processRoles(req, res, mapping) in your extended class if you want to query the LDAP server for the security authorization etc.

**ActionServlet:** You can extend the ActionServlet class if you want to execute your business logic at the application startup or shutdown or during individual request processing. You should take this approach only when the above mentioned approaches are not feasible.

**Q 116:** What design patterns are used in Struts? **DP**

**A 116:** Struts is based on model 2 MVC (Model-View-Controller) architecture. Struts controller uses the **command design pattern** (Refer Q11 in How would you go about section) and the action classes use the **adapter design pattern**. The process() method of the RequestProcessor uses the **template method design pattern** (Refer Q11 in How would you go about section). Struts also implement the following J2EE design patterns

- Service to Worker (Refer Q25 in Enterprise section).
- Dispatcher View (Refer Q25 in Enterprise section).
- Composite View (Struts Tiles) (Refer Q25 in Enterprise section)
- Front Controller (Refer Q24 in Enterprise section).
- View Helper (Refer Q25 in Enterprise section).
- Synchronizer Token (Refer Q111 in Enterprise section).

### Enterprise - Web and Application servers

**Q 117:** What application servers, Web servers, LDAP servers, and Database servers have you used?

**A 117:**

<b>Web Servers</b>	Apache, Microsoft IIS, Netscape, Domino etc
<b>Application Servers</b>	IBM Websphere, BEA Weblogic, Apache Tomcat, Borland Enterprise Server, Fujitsu Interstage, JBoss, ATG Dynamo etc
<b>LDAP Servers</b>	IPlanet's directory server, SiemensDirX etc
<b>Database Servers</b>	IBM DB2, Oracle, SQL Server, Sybase, Informix

**Q 118:** What is the difference between a Web server and an application server? **SF**

**A 118:** In general, an application server prepares data for a Web server -- for example, gathering data from databases, applying relevant business rules, processing security checks, and/or storing the state of a user's session. The term application server may be misleading since the functionality isn't limited to applications. Its role is more as retriever and manager of data and processes used by anything running on a Web server. In the coming age of Web services, application servers will probably have an even more important role in managing service oriented components. One of the reasons for using an application server is to improve performance by off-loading tasks from a Web server. When heavy traffic has more users, more transactions, more data, and more security checks then more likely a Web server becomes a bottleneck.

<b>Web Server</b>	<b>Application Server</b>
Supports HTTP protocol. When a Web server receives an HTTP request, it responds with an HTTP response, such as sending back an HTML page (static content) or delegates the dynamic response generation to some other program such as CGI scripts or Servlets or JSPs in an application server.	Exposes <b>business logic</b> and <b>dynamic content</b> to a client through various protocols such as HTTP, TCP/IP, IIOP, JRMP etc.
Uses various scalability and fault-tolerance techniques.	Uses various scalability and fault-tolerance techniques. In addition provides resource pooling, component life cycle management, transaction management, messaging, security etc.

**Q 119:** What is a virtual host? **SF**

**A 119:** The term virtual host refers to the practice of maintaining **more than one server on one machine**. They are differentiated by their host names. You can have name based virtual hosts and IP address based virtual hosts. For example

A name-based "virtual host" has a **unique domain name**, but the same IP address. For example, www.company1.com and www.company2.com can have the same IP address 192.168.0.10 and share the same Web server. We can configure the Web server as follows:

```
NameVirtualHost 192.168.0.10

<VirtualHost 192.168.0.10>
 ServerName www.company1.com
 DocumentRoot /web/company1
</VirtualHost>

<VirtualHost 192.168.0.10>
 ServerName www.company2.com
 DocumentRoot /web/company2
</VirtualHost>
```

In this scenario, both www.company1.com and www.company2.com are registered with the standard **domain name service (DNS)** registry as having the IP address 192.168.0.10. A user types in the URL http://www.company1.com/hello.jsp in their browser. The user's computer resolves the name

[www.company1.com](http://www.company1.com) to the IP address 192.168.0.10. The Web server on the machine that has the IP address 192.168.0.10, so it receives the request. The Web server determines which virtual host to use by matching the request URL it gets from an HTTP header submitted by the browser with the "ServerName" parameter in the configuration file shown above.

Name-based virtual hosting is usually easier, since you have to only configure your DNS server to map each hostname to a single IP address and then configure the Web server to recognize the different hostnames as discussed in the previous paragraph. Name-based virtual hosting also eases the demand for scarce IP addresses limited by physical network connections [but modern operation systems supports use of virtual interfaces, which are also known as IP aliases]. Therefore you should use name-based virtual hosting unless there is a specific reason to choose IP-based virtual hosting. Some reasons why you might consider using IP-based virtual hosting:

- Name-based virtual hosting cannot be used with SSL based secure servers because of the nature of the SSL protocol.
- Some operating systems and network equipment implement bandwidth management techniques that cannot differentiate between hosts unless they are on separate IP addresses.
- IP based virtual hosts are useful, when you want to manage more than one site (like live, demo, staging etc) on the same server where hosts inherit the characteristics defined by your main host. But when using SSL for example, a unique IP address is necessary.

For example in development environment when using the test client and the server on the same machine we can define the host file as shown below:

**UNIX user:** /etc/hosts

**Windows user:** C:\WINDOWS\SYSTEM32\DRIVERS\ETC\HOSTS

127.0.0.1	localhost
127.0.0.1	www.company1.com
127.0.0.1	www.company2.com

[Reference: <http://httpd.apache.org/docs/1.3/vhosts/>]

**Q 120:** What is application server clustering? **S1**

**A 120:** An application server cluster consists of a number of application servers loosely coupled on a network. The server cluster or server group is generally distributed over a number of machines or nodes. The important point to note is that the cluster appears as a single server to its clients.

The goals of application server clustering are:

- **Scalability:** should be able to add new servers on the existing node or add new additional nodes to enable the server to handle increasing loads without performance degradation, and in a manner transparent to the end users.
- **Load balancing:** Each server in the cluster should process a fair share of client load, in proportion to its processing power, to avoid overloading of some and under utilization of other server resources. Load distribution should remain balanced even as load changes with time.
- **High availability:** Clients should be able to access the server at almost all times. Server usage should be transparent to hardware and software failures. If a server or node fails, its workload should be moved over to other servers, automatically as fast as possible and the application should continue to run uninterrupted. This method provides a fair degree of application system fault-tolerance. After failure, the entire load should be redistributed equally among working servers of the system.

[Good read: Uncover the hood of J2EE clustering by Wang Yu on <http://www.theserverside.com> ]

**Q 121:** Explain Java Management Extensions (**JMX?**) **SF**

**A 121:** JMX framework can improve the manageability of your application by

- Monitoring your application for performance problems, critical events, error condition statistics, etc. **For example** you can be notified if there is a sudden increase in traffic or sudden drop in performance of your website.
  - Making your application more controllable and configurable at runtime by directly exposing application API and parameters. **For example** you could switch your database connection to an alternate server. You can also change the level of debugging and logging within the application without stopping the server.
  - By interfacing JMX to your hardware, database server and application server, health checks can be performed of your infrastructure.
- 

**Q 122:** Explain some of the portability issues between different application servers? **S1**

**A 122:** Transaction isolation levels, lazy loading and dirty marker strategies for EJB, class loading visibility etc.

#### Enterprise - Best practices and performance considerations

**Q 123:** Give some tips on J2EE application server performance tuning? **P1**

**A 123:**

- Set the Web container threads, which will be used to process incoming HTTP requests. The minimum size should be tuned to handle the average load of the container and maximum should be tuned to handle the peak load. The maximum size should be less than or equal to the number of threads in your Web server.
  - When an EJB is called from a servlet or another EJB within the same JVM (i.e. same application server) then performance can be improved by running EJBs in pass-by-reference mode as oppose to pass-by-value which is the default mode. Care should be taken to test the application properly before going into production because some valid applications may not work correctly when pass-by-reference setting is switched on.
  - Application servers maintain a pool of JDBC resources so that a new connection does not need to be created for each transaction. Application servers can also cache your prepared statements to improve performance. So you can tune the minimum and maximum size of these pools.
  - Tune your initial heap size for the JVM so that the garbage collector runs at a suitable interval so that it does not cause any unnecessary overhead. Adjust the value as required to improve performance.
  - Set the session manager settings appropriately based on following guidelines:
    - Set the appropriate value for in memory session count.
    - Reduce the session size.
    - Don't enable session persistence unless required by your application.
    - Invalidate your sessions when you are finished with them by setting appropriate session timeout.
  - Calls to EJB from a separate JVM are handled by ORB (Object Request Broker). ORB uses a pool of threads to handle these requests. The thread pool size should be set appropriately to handle average and peak loads.
  - If a servlet or JSP file is called frequently with identical URL parameters then they can be dynamically cached to improve performance.
  - Turn the application server tracing off unless required for debugging.
  - Some application servers support lazy loading and dirty marker strategies with EJB to improve performance.
- 

**Q 124:** Explain some of the J2EE best practices? **BP**

**A 124:**

- **Recycle your valuable resources by either pooling or caching.** You should create a limited number of resources and share them from a common pool (e.g. pool of threads, pool of database connections, pool of objects etc). Caching is simply another type of pooling where instead of pooling a connection or object, you are pooling remote data (database data) and placing it in the memory (using Hashtable etc).
- **Avoid embedding business logic in a protocol dependent manner** like in JSPs, HttpServlets, Struts action classes etc. This is because your business logic should be not only executed by your Web clients but also required to be shared by various GUI clients like Swing based stand alone application, WAP clients etc.
- **Automate** the build process with tools like **Ant**, **CruiseControl**, and **Maven** etc. In an enterprise application the build process can become quite complex and confusing.
- **Build** test cases first (i.e. Test Driven Development (TDD), refer section Emerging Technologies) using tools like **JUnit**. Automate the testing process and integrate it with build process.
- **Separate HTML code from the Java code:** Combining HTML and Java code in the same source code can make the code less readable. Mixing HTML and scriptlet will make the code extremely difficult to read and maintain. The display or behaviour logic can be implemented as a custom tags by the Java developers and Web designers can use these Tags as the ordinary XHTML tags.
- It is best practice to use multi-threading and stay away from **single threaded model of the servlet** unless otherwise there is a compelling reason for it. Shared resources can be synchronized or used in read-only manner or shared values can be stored in a database table. Single threaded model can adversely affect performance.
- **Apply the following JSP best practices:**
  - **Place data access logic in JavaBeans:** The code within the JavaBean is readily accessible to other JSPs and Servlets.
  - **Factor shared behaviour out of Custom Tags into common JavaBeans classes:** The custom tags are not used outside JSPs. To avoid duplication of behaviour or business logic, move the logic into JavaBeans and get the custom tags to utilize the beans.
  - **Choose the right “*include*” mechanism:** What are the differences between static and a dynamic include? Using includes will improve code reuse and maintenance through modular design. Which one to use? Refer **Q31** in Enterprise section.
  - **Use style sheets** (e.g. css), **template mechanism** (e.g. struts tiles etc) and **appropriate comments** (both hidden and output comments).
  - If you are using EJBs apply the EJB best practices as described in **Q82** in Enterprise section.
  - Use the J2EE standard packaging specification to improve portability across Application Servers.
  - Use proven frameworks like **Struts**, **Spring**, **Hibernate**, **JSF** etc.
  - Apply appropriate proven J2EE design patterns to improve performance and minimise network communications cost (Session façade pattern, Value Object pattern etc).
  - Batch database requests to improve performance. For example

```
Connection con = DriverManager.getConnection(".....");
Statement stmt = con.createStatement();
stmt.addBatch("INSERT INTO Address.....");
stmt.addBatch("INSERT INTO Contact.....");
stmt.addBatch("INSERT INTO Personal");
int[] countUpdates = stmt.executeBatch();
```

Use “**PreparedStatements**” instead of ordinary “Statements” for repeated reads.

- Avoid resource leaks by
  - Closing all database connections after you have used them.
  - Clean up objects after you have finished with them especially when an object having a long life cycle refers to a number of objects with short life cycles (you have the potential for memory leak).

- Poor exception handling where the connections do not get closed properly and clean up code that never gets called. You should put clean up code in a **finally {}** block.
- Handle and propagate exceptions correctly. Decide between checked and unchecked (i.e RunTime exceptions) exceptions.

---

**Q 125:** Explain some of the J2EE best practices to improve performance? **BP PI**

**A 125:** In short manage valuable resources wisely and recycle them where possible, minimise network overheads and serialization cost, and optimise all your database operations.

- **Manage and recycle your valuable resources by either pooling or caching.** You should create a limited number of resources and share them from a common pool (e.g. pool of threads, pool of database connections, pool of objects etc). Caching is simply another type of pooling where instead of pooling a connection or object, you are pooling remote data (database data), and placing it in memory (using Hashtable etc). Unused stateful session beans must be removed explicitly and appropriate idle timeout should be set to control stateful session bean life cycle.
- **Use effective design patterns to minimise network overheads** (Session facade, Value Object etc Refer **Q84, Q85** in Enterprise section), use of fast-lane reader pattern for database access (Refer **Q86** in Enterprise section). Caching of retrieved JNDI InitialContexts, factory objects (e.g. EJB homes) etc. using the service locator design pattern, which reduces expensive JNDI access with the help of caching strategies.
- **Minimise serialization costs** by marking references (like file handles, database connections etc), which do not require serialization by declaring them '**transient**' (Refer **Q19** in Java section). Use pass-by-reference where possible as opposed to pass by value.
- **Set appropriate timeouts:** for the HttpSession objects, after which the session expires, set idle timeout for stateful session beans etc.
- Improve the **performance of database operations** with the following tips:
  - Database connections should be released when not needed anymore, otherwise there will be potential resource leakage problems.
  - Apply least restrictive but valid transaction isolation level.
  - Use JDBC prepared statements for overall database efficiency and for batching repetitive inserts and updates. Also batch database requests to improve performance.
  - When you first establish a connection with a database by default it is in auto-commit mode. For better performance turn auto-commit off by calling the connection.setAutoCommit(false) method.
  - Where appropriate (you are loading 100 objects into memory but use only 5 objects) **lazy load** your data to avoid loading the whole database into memory using the virtual proxy pattern. Virtual proxy is an object, which looks like an object but actually contains no fields until when one of its methods is called does it load the correct object from the database.
  - Where appropriate **eager load** your data to avoid frequently accessing the database every time over the network.

---

#### Enterprise – Logging, testing and deployment

---

**Q 126:** Give an overview of log4J? **SF**

**A 126:** Log4j is a logging framework for Java. Log4J is designed to be fast and flexible. Log4J has 3 main components which work together to enable developers to log messages:

- Loggers [was called *Category* prior to version 1.2]
- Appenders
- Layout

**Logger:** The foremost advantage of any logging API like log4J, apache commons logging etc over plain System.out.println is its ability to disable certain log statements while allowing others to print unhindered. Loggers are hierarchical. The root logger exists at the top of the hierarchy. The root logger always exists and it cannot be retrieved by name. The hierarchical nature of the logger is denoted by “.” notation. For example the logger “java.util” is the parent of child logger “java.util.Vector” and so on. Loggers may be assigned priorities such as DEBUG, INFO, WARN, ERROR and FATAL. If a given logger is not assigned a priority, then it inherits the priority from its closest ancestor. The logging requests are made by invoking one of the following printing methods of the logger instance: debug(), info(), warn(), error(), fatal().

**Appenders and Layouts:** In addition to selectively enabling and disabling logging requests based on the logger, the log4J allows logging requests to multiple destinations. In log4J terms the output destination is an appender. There are appenders for console, files, remote sockets, JMS, etc. One logger can have more than one appender. A logging request for a given logger will be forwarded to all the appenders in that logger plus the other appenders higher in the hierarchy. In addition to the output destination the output format can be categorised as well. This is accomplished by associating layout with an appender. The layout is responsible for formatting the logging request according to user's settings.

#### Sample configuration file:

```
#set the root logger priority to DEBUG and its appender to App1
log4j.rootLogger=DEBUG, App1

#App1 is set to a console appender
log4j.appender.App1=org.apache.log4j.ConsoleAppender

#appender App1 uses a pattern layout
log4j.appender.App1.layout=org.apache.log4j.PatternLayout
log4j.appender.App1.layout.ConversionPattern=%-4r [%t] %-5p %c %x -%m%n

Print only messages of priority WARN or above in the package com.myapp
log4j.Logger.com.myapp=WARN
```

XML configuration for log4j is available, and is usually the best practise.

**Q 127:** How do you initialize and use Log4J? **SF CO**

**A 127:**

```
public class MyApp {
 //Logger is a utility wrapper class to be written with appropriate printing methods
 static Logger log = Logger.getLogger (MyApp.class.getName());

 public void my method() {
 if(log.isDebugEnabled())
 log.debug("This line is reached....." + var1 + "-" + var2);
 }
}
```

**Q 128:** What is the hidden cost of parameter construction when using Log4J? **SF PI**

**A 128:**

**Do not use in frequently accessed methods or loops:** **CO**

```
log.debug ("Line number" + intVal + " is less than " + String.valueOf(array[i]));
```

The above construction has a **performance cost in frequently accessed methods and loops** in constructing the message parameter, concatenating the *String* etc regardless of whether the message will be logged or not.

**Do use in frequently accessed methods or loops:** **CO**

```
If (log.isDebugEnabled()) {
 log.debug ("Line number" + intVal + " is less than " + String.valueOf(array[i]));
}
```

The above construction will avoid the parameter construction cost by only constructing the message parameter when you are in debug mode. But it is not a best practise to place `log.isDebugEnabled()` around all debug code.

---

**Q 129:** What is the test phases and cycles? **[SD]**

**A 129:**

- **Unit tests** (e.g. JUnit etc, carried out by developers).  
There are two popular approaches to testing server-side classes: **mock objects**, which test classes by simulating the server container, and **in-container** testing, which tests classes running in the actual server container. If you are using Struts framework, *StrutsTestCase* for JUnit allows you to use either approach, with very minimal impact on your actual unit test code.
- **System tests or functional tests** (carried out by business analysts and/or testers).
- **Integration tests** (carried out by business analysts, testers, developers etc).
- **Regression tests** (carried out by business analysts and testers).
- **Stress volume tests or load tests** (carried out by technical staff).
- **User acceptance tests** (UAT – carried out by end users).

Each of the above test phases will be carried out in cycles. Refer **Q13** in How would you go about... section for JUnit, which is an open source unit-testing framework.

---

**Q 130:** Brief on deployment environments you are familiar with?

**A 130:** Differ from project team to project team **[Hint]** :

**Application environments where “ear” files get deployed.**

**Development box:** can have the following instances of environments in the same machine (need not be clustered).

- Development environment → used by developers.
- System testing environment → used by business analysts.

**Staging box:** can have the following instances of environments in the same machine (preferably clustered servers with load balancing)

- Integration testing environment → used for integration testing, user acceptance testing etc.
- Pre-prod environment → used for user acceptance testing, regression testing, and load testing or stress volume testing (SVT). [This environment should be exactly same as the production environment].

**Production box:**

- Production environment → live site used by actual users.

**Data environments (Database)**

**Note:** Separate boxes [not the same boxes as where applications (i.e. ear files) are deployed]

- **Development box** (database).  
Used by applications on development and system testing environments. Separate instances can be created on the same box for separate environments like development and system testing.
- **Staging Box** (database)  
Used by applications on integration testing and user acceptance testing environments. Separate instances can be created on the same box for separate environments.
- **Production Box** (database)  
Live data used by actual users of the system.

**Enterprise - Personal**

**Q 131:** Tell me about yourself or about some of the recent projects you have worked with? What do you consider your most significant achievement? Why do you think you are qualified for this position? Why should we hire you and what kind of contributions will you make?

**A 131:** [Hint:] Pick your recent projects and give a brief overview of it. Also it is imperative that during your briefing that you demonstrate how you applied your skills and knowledge in some of the following key areas and fixed any issues.

- **Design Concepts:** Refer Q02, Q03, Q19, Q20, Q21, Q91, Q98, and Q101.
- **Design Patterns:** Refer Q03, Q24, Q25, Q83, Q84, Q85, Q86, Q87, Q88 and Q111.
- **Performance issues:** Refer Q10, Q16, Q45, Q46, Q97, Q98, Q100, Q123, and Q125.
- **Memory issues:** Refer Q45 and Q93
- **Multi-threading (Concurrency issues):** Refer Q16, Q34, and Q113
- **Exception Handling:** Refer Q76 and Q77
- **Transactional issues:** Refer Q43, Q71, Q72, Q73, Q74, Q75 and Q77.
- **Security issues:** Refer Q23, Q58, and Q81
- **Scalability issues:** Refer Q20, Q21, Q120 and Q122.
- **Best practices:** Refer Q10, Q16, Q39, Q40, Q46, Q82, Q124, and Q125

Refer Q66 – Q72 in Java section for frequently asked non-technical questions.

**Q 132:** Have you used any load testing tools?

**A 132:** Rational Robot, JMeter, LoadRunner, etc.

**Q 133:** What source control systems have you used? **SD**

**A 133:** CVS, VSS (Visual Source Safe), Rational clear case etc. Refer Q13 in How would you go about section.... for CVS.

**Q 134:** What operating systems are you comfortable with? **SD**

**A 134:** NT, Unix, Linux, Solaris etc

**Q 135:** Which on-line technical resources do you use to resolve any design and/or development issues?

**A 135:** <http://www.theserverside.com>, <http://www.javaworld.com>, <http://www-136.ibm.com/developerworks/Java/>, <http://java.sun.com/>, [www.javaperformancetuning.com](http://www.javaperformancetuning.com) etc

**Enterprise – Software development process**

**Q 136:** What software development processes/principles are you familiar with? **SD**

**A 136:** Agile (i.e. **lightweight**) software development process is gaining popularity and momentum across organizations.

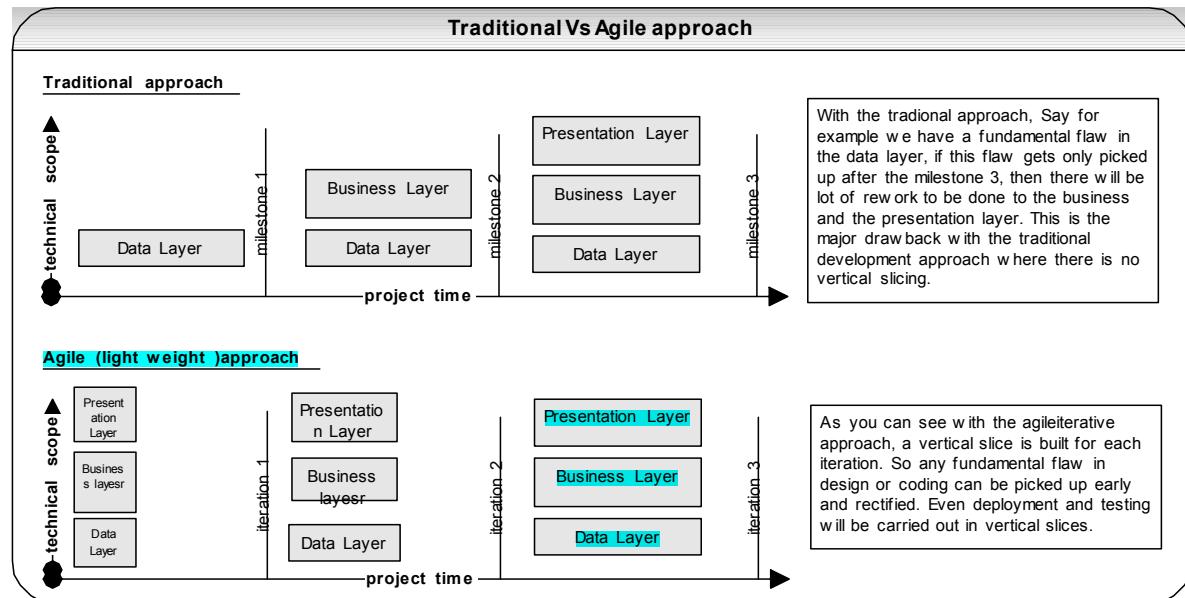
**Agile software development manifesto** → [Good read: <http://www.agilemanifesto.org/principles.html>].

- Highest priority is to satisfy the customer.
- Welcome requirement changes even late in development life cycle.
- Business people and developers should work collaboratively.
- Form teams with motivated individuals who produce best designs and architectures.

agile methodology is described as “iterative” and “incremental.” every aspect of development — requirements, design, etc. — is continually revisited. When a team stops and re-evaluates the direction of a project every two weeks, there’s time to steer it in another direction.

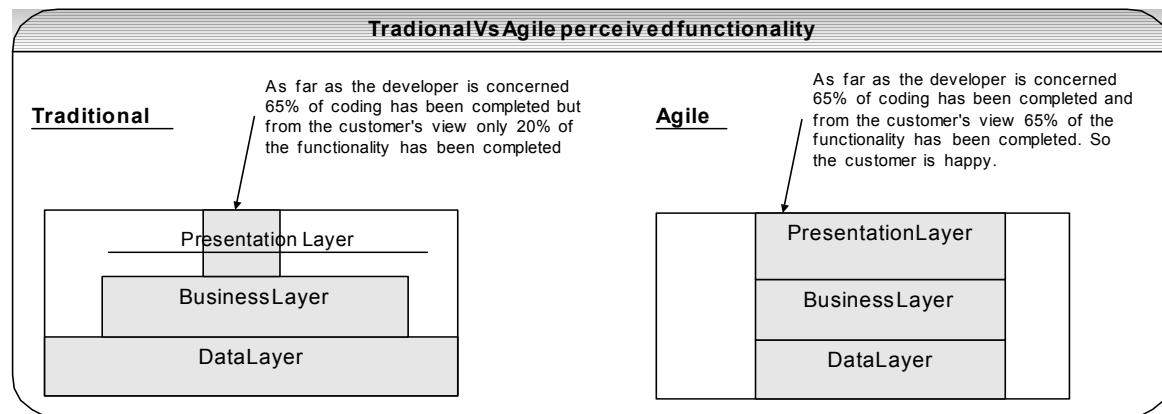
- Teams should be pro-active on how to become more effective without becoming complacent.
- Quality working software is the primary measure of progress.

**Why is iterative development with vertical slicing used in agile development?** Your overall software quality can be improved through iterative development, which provides you with constant feedback.



Several methodologies fit under this agile development methodology banner. All these methodologies share many characteristics like **iterative and incremental development, test driven development, stand up meetings to improve communication, automatic testing, build and continuous integration of code** etc. Among all the agile methodologies XP is the one which has got the most attention. Different companies use different flavours of agile methodologies by using different combinations of methodologies.

**How does vertical slicing influence customer perception?** With the iterative and incremental approach, customer will be comfortable with the progress of the development as opposed to traditional big bang approach.



- EXtreme Programming [XP]** → simple design, pair programming, unit testing, refactoring, collective code ownership, coding standards, etc. Refer **Q10** in “How would you go about...” section. XP has four key values: Communication, Feedback, Simplicity and Courage. It then builds up some tried and tested practices and techniques. XP has a strong emphasis on testing where tests are integrated into continuous integration and build process, which yields a highly stable platform. XP is designed for smaller teams of 20 – 30 people.
- RUP (Rational Unified Process)** → Model driven architecture, design and development; customizable frameworks for scalable process; iterative development methodology; Re-use of architecture, code, component, framework, patterns etc. RUP can be used as an agile process for smaller teams of 20-30

people, or as a heavy weight process for larger teams of 50-100 people. Refer **Q103 – Q105** in Enterprise section.

- **Feature Driven Development [FDD]** → Jeff De Luca and long time OO guru Peter Coad developed feature Driven Development (FDD). Like the other **adaptive methodologies**, it focuses on short iterations that deliver tangible functionality. FDD was originally designed for larger project teams of around 50 people. In FDD's case the iterations are two weeks long. FDD has five processes. The first three are done at the beginning of the project. The last two are done within each iteration.
  1. Develop an Overall Model
  2. Build a Features List
  3. Plan by Feature
  4. Design by Feature
  5. Build by Feature

The developers come in two kinds: class owners and chief programmers. The chief programmers are the most experienced developers. They are assigned features to be built. However they don't build them alone. Instead the chief programmer identifies which classes are involved in implementing the feature and gathers their class owners together to form a feature team for developing that feature. The chief programmer acts as the coordinator, lead designer, and mentor while the class owners do much of the coding of the feature.

- **Test Driven Development [TDD]** → TDD is an iterative software development process where you **first write the test with the idea that it must fail**. Refer **Q1** in Emerging Technologies/Frameworks section...
- **Scrum** → Scrum divides a project into sprints (aka iterations) of 30 days. Before you begin a sprint you define the functionality required for that sprint and leave the team to deliver it. But every day the team holds a short (10 – 15 minute) meeting, called a scrum where the team runs through what it will achieve in the next day. Some of the questions asked in the scrum meetings are:
  - What did you do since the last scrum meetings?
  - Do you have any obstacles?
  - What will you do before next meeting?

This is very similar to stand-up meetings in XP and iterative development process in RUP.

### Enterprise – Key Points

- J2EE is a **3-tier** (or **n-tier**) system. Each tier is logically separated and loosely coupled from each other, and may be distributed.
- J2EE applications are developed using **MVC architecture**, which divides the functionality of displaying and maintaining of the data to minimise the degree of coupling between enterprise components.
- J2EE modules are deployed as ear, war and jar files, which are standard application deployment archive files.
- HTTP is a stateless protocol and state can be maintained between client requests using HttpSession, URL rewriting, hidden fields and cookies. HttpSession is the recommended approach.
- Servlets and JSPs are by default multi-threaded, and care should be taken in declaring instance variables and accessing shared resources. It is possible to have a single threaded model of a servlet or a JSP but this can adversely affect performance.
- Clustering promotes high availability and scalability. The considerations for servlet clustering are:
  - Objects stored in the session should be serializable.
  - Design for idempotence.
  - Avoid using instance and static variables in read and write mode.
  - Avoid storing values in the *ServletContext*.
  - Avoid using `java.io.*` and use `getResourceAsStream()` instead.

- JSPs have a translation or a compilation process where the JSP engine translates and compiles a JSP file into a JSP servlet.
- JSPs have 4 different **scope** values: page, request, session and application. JSPs can be included **statically**, where all the included JSP pages are compiled into a single servlet during the translation or compilation phase or included **dynamically**, where included JSPs are compiled into separate servlets and the content generated by these servlets are included at runtime in the JSP response.
- Avoid scriptlet code in your JSPs and use **JavaBeans** or **custom tags** (e.g. Struts tags, JSTL tags, JSF tags etc) instead.
- Databases can run out cursors if the connections are not closed properly. The valuable resources like connections and statements should be enclosed in a **try{}** and **finally{}** block.
- Prepared statements offer better performance as opposed to statements, as they are **precompiled** and **reuse the same execution plan** with different arguments. Prepared statements are also more secure because they use bind variables, which can prevent SQL injection attacks.
- JNDI provides a generic interface to LDAP and other directory services like NDS, DNS etc.
- In your code always make use of a **logical JNDI reference** (`java:comp/env/ejb/MyBean`) as opposed to **physical JNDI reference** (`ejb/MyBean`) because you cannot guarantee that the physical JNDI location you specify in your code will be available. Your code will break if the physical location is changed.
- LDAP servers are typically used in J2EE applications to authenticate and authorise users. LDAP servers are hierarchical and are **optimized for read access**, so likely to be faster than database in providing read access.
- RMI facilitates object method calls between JVMs. JVMs can be located on separate host machines, still one JVM can invoke methods belonging to an object residing in another JVM (i.e. address space). RMI uses object serialization to marshal and unmarshal parameters. The remote objects should extend the `UnicastRemoteObject`.
- To go through a firewall, the RMI protocol can be embedded within the firewall trusted HTTP protocol, which is called **HTTP tunnelling**.
- EJB (i.e. 2.x) is a remote, distributed multi-tier system, which supports protocols like JRMP, IIOP, and HTTP etc. EJB components contain business logic and system level supports like security, transaction, instance pooling, multi-threading, object life-cycles etc are managed by the EJB container and hence simplify the programming effort. Having said this, there are emerging technologies like:
  - Hibernate, which is an open source object-to-relational (O/R) mapping framework.
  - EJB 3.0, which is taking ease of development very seriously and has adjusted its model to offer the plain old Java objects (i.e. POJOs) based persistence and the new O/R mapping model based on hibernate.

Refer **Q14 – Q18** in Emerging technologies / Frameworks section for brief discussion on hibernate and EJB 3.0.

- EJB transaction attributes (like Required, Mandatory, RequiresNew, Supports etc) are specified declaratively through EJB deployment descriptors. Isolation levels are not part of the EJB 2.x specification. So the isolation levels can be set on the resource manager either explicitly on the *Connection* or via the application server specific configuration.
- A transaction is often described by **ACID** (Atomic, Consistent, Isolated and Durable) properties. A **distributed transaction** is an ACID transaction between two or more independent transactional resources like two separate databases. A **2-phase commit** is an approach for committing a distributed transaction in 2 phases.
- EJB 2.x has two types of exceptions:
  - **System exception:** is an unchecked exception derived from `java.lang.RuntimeException`. It is thrown by the system and is not recoverable.
  - **Application exception:** is specific to an application and is thrown because of violation of business rules.
- EJB container managed transactions are automatically rolled back when a system exception occurs. This is possible because the container can intercept system exceptions. However when an application exception occurs, the container does not intercept and leaves it to the code to roll back using `ctx.setRollbackOnly()` method.
- EJB containers can make use of **lazy loading** (i.e. not creating an object until it is accessed) and **dirty marker** (ie persist only the entity beans that have been modified) strategies to improve entity beans performance.

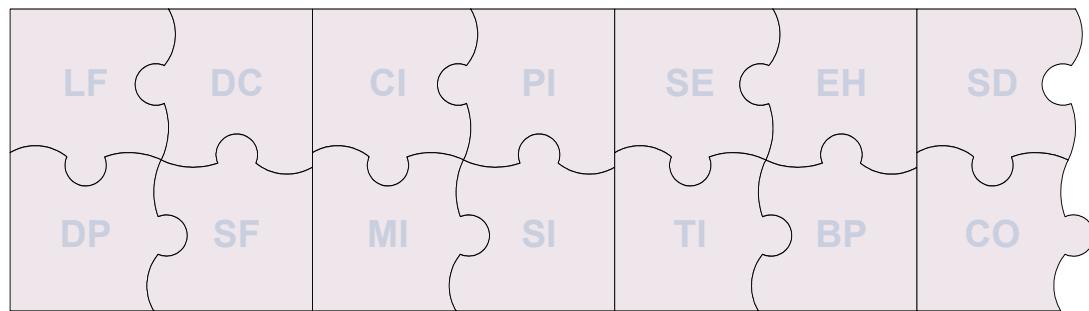
- Message Oriented Middleware (MOM) is a software infrastructure that asynchronously communicates with other disparate systems through the production and consumption of messages. Messaging enables loosely coupled distributed communication. Java Messaging Service (**JMS**) is a Java API that allows applications to create, send, receive read messages in a standard way, hence improves portability.
- Some of the design decisions you need to make in JMS are message acknowledgement modes, transaction modes, delivery modes etc, synchronous vs. asynchronous paradigm, message body types, setting appropriate timeouts etc.
- XML documents can be processed in your Java/J2EE application either using a SAX parser, which is event driven or a DOM parser, which creates a tree structure in memory. The other XML related technologies are DTD, XSD, XSL, XPath, etc and Java and XML based technologies are JAXP, JAXB etc.
- There is an impedance mismatch between object and relational technology. Classes represent both data and behaviour whereas relational database tables just implement data. Inheritance class structure can be mapped to relational data model in one of the following ways:
  - Map class hierarchy to single database table.
  - Map each class to its own table.
  - Map each concrete class to its own table
  - Generic meta-data driven approach.
- Normalize data in your database for accuracy and denormalize data in your database for performance.
- **RUP** (Rational Unified Process) has 4 phases in the following order Inception, Elaboration, Construction, and Transition. Agile (i.e. lightweight) software development process is gaining popularity and momentum across organizations. Several methodologies like XP, RUP, Scrum, FDD, TDD etc fit under this agile development methodology banner. All these methodologies share many characteristics like iterative and incremental development, stand-up meetings to improve communication, automatic build, testing and continuous integration etc.
- UML is applicable to the object oriented (OO) problem solving. There are different types of UML diagrams like use case diagrams, class diagrams, sequence diagrams, collaboration diagrams, state chart diagrams, activity diagrams, component diagrams, deployment diagrams etc.
- Class diagrams are vital within OO methods. Class diagrams have the following possible relationships, association, aggregation, composition, generalization, and dependency.
- Struts is an MVC framework. Struts action classes are not thread-safe and care should be taken in declaring instance variables or accessing other shared resources. JSF is another Web UI framework like Struts gaining popularity and momentum.
- Log4j has three main components: *loggers*, *appenders* and *layouts*. Logger is a utility wrapper class. JUnit is an open source unit-testing framework.
- You can improve the performance of a J2EE application as follows :
  1. Manage and recycle your valuable resources like connections, threads etc by either pooling or caching.
  2. Use effective design patterns like session façade, value object, fast lane reader etc to minimise network overheads.
  3. Set appropriate timeouts for HttpSession objects.
  4. Use JDBC prepared statements as opposed to statements.
  5. Release database connections in a finally {} block when finished.
  6. Apply least restrictive but valid transaction isolation level.
  7. Batch database requests.
  8. Minimise serialization costs by marking references like file handles, database connections, etc which do not require serialization by declaring them transient.
- Some of the J2EE best practices are:
  1. Recycle your valuable resources by either pooling or caching.
  2. Automate your build process with tools like Ant, CruiseControl, and Maven etc, and continuously integrate your code into your build process.
  3. Build test cases first using tools like JUnit.
  4. Use standard J2EE packaging to improve portability.
  5. Apply appropriate proven design patterns.

- 6. Use proven frameworks like Struts, Spring, Hibernate, JSF, JUnit, Log4J, etc.
  - 7. Handle and propagate exceptions correctly.
  - 8. Avoid resource leaks by closing all database connections after you have used them.
- The goals of application server clustering are to achieve scalability, load balancing, and high availability.
  - Java Management Extension (JMX) framework can improve the manageability of your application, for performance problems, critical events, error conditions etc and perform health checks on your hardware, database server etc. You can also configure and control your application at runtime.
  - Finally get familiarised with some of the key Java & J2EE **design patterns** like:
    1. **MVC design pattern:** J2EE uses this design pattern or architecture.
    2. **Chain of responsibility design pattern:** Servlet filters use a slightly modified version of chain of responsibility design pattern.
    3. **Front controller J2EE design pattern:** provides a centralized access point for HTTP request handling to support the integration system services like security, data validation etc. This is a popular J2EE design pattern.
    4. **Composite view J2EE design pattern:** creates an aggregate view from atomic sub-views.
    5. **View helper J2EE design pattern:** avoids duplication of code. The helper classes are JavaBeans and custom tags (e.g. Struts tags, JSF tags, JSTL tags etc).
    6. **Service to worker and dispatcher view J2EE design pattern:** These two patterns are a combination of front controller and view helper patterns with a dispatcher component. These two patterns differ in the way they suggest different division of responsibility among components.
    7. **Bridge design pattern:** Java Data Base Connectivity (JDBC) uses the bridge design pattern. The JDBC API provides an abstraction and the JDBC drivers provide the implementation.
    8. **Proxy design pattern:** RMI & EJB uses the proxy design pattern. A popular design pattern.
    9. **Business delegate J2EE design pattern:** used to reduce the coupling between the presentation tier and the business services tier components.
    10. **Session façade J2EE design pattern:** too many fine-grained method calls between the client and the server will lead to network overhead and tight coupling. Use a session bean as a façade to provide a coarse-grained service access layer to clients.
    11. **Value object J2EE design pattern:** avoid fine-grained method calls by creating a value object, which will help the client, make a coarse-grained call.
    12. **Fast-lane reader J2EE design pattern:** access the persistence layer directly using a DAO (Data Access Object) pattern instead of using entity beans.
    13. **Service locator J2EE design pattern:** expensive and redundant JNDI lookups can be avoided by caching and reusing the already looked up service objects.

**Recommended reading on J2EE design patterns:**

- Core J2EE Patterns: Best Practices and Design Strategies, Second Edition (Hardcover) by Deepak Alur, Dan Malks, John Crupi.

**Let us put all together in  
the next section**



## SECTION THREE

### How would you go about...?

- This section basically assesses your knowledge of how to perform certain tasks like documenting your project, identifying any potential performance, memory, transactional, and/or design issues etc.
- It also assesses if you have performed any of these tasks before. If you have not done a particular task, you can demonstrate that you know how to go about it if the task is assigned to you.
- This section also recaps some of the key considerations discussed in the **Java** and **Enterprise** sections. Question numbers are used for cross-referencing with **Java** and **Enterprise** sections.
- **Q11 & Q13** are discussed in more detail and can be used as a quick reference guide in a software project. All the other questions excluding **Q11 & Q13** can be read just before an interview.

**Q 01:** How would you go about documenting your Java/J2EE application?

**A 01:** To be successful with a Java/J2EE project, proper documentation is vital.

- Before embarking on coding get the business requirements down. Build a complete list of requested features, sample screen shots (if available), use case diagrams, business rules etc as a **functional specification** document. This is the phase where business analysts and developers will be asking questions about user interface requirements, data tier integration requirements, use cases etc. Also prioritize the features based on the business goals, lead-times and iterations required for implementation.
- Prepare a **technical specification** document based on the functional specification. The technical specification document should cover:
  - ❖ **Purpose of the document:** This document will emphasise the customer service functionality ...
  - ❖ **Overview:** This section basically covers background information, scope, any inclusions and/or exclusions, referenced documents etc.
  - ❖ **Basic architecture:** discusses or references baseline architecture document. Answers questions like Will it scale? Can this performance be improved? Is it extendable and/or maintainable? Are there any security issues? Describe the vertical slices to be used in the early iterations, and the concepts to be proved by each slice. Etc. **For example** which MVC [model-1, model-2 etc] paradigms (Refer Q3 in Enterprise section for MVC) should we use? Should we use Struts, JSF, Spring etc or build our own framework? Should we use a business delegate (Refer Q83 in Enterprise section for business delegate) to decouple middle tier with the client tier? Should we use AOP (Aspect Oriented Programming) (Refer Q3 in Emerging Technologies/Frameworks)? Should we use dependency injection? Should we use annotations? Do we require internationalization? Etc.
  - ❖ **Assumptions, Dependencies, Risks and Issues:** highlight all the assumptions, dependencies, risks and issues. For example list all the risks you can identify.
  - ❖ **Design alternatives** for each key functional requirement. Also discuss why a particular design alternative was chosen over the others. This process will encourage developers analyse the possible design alternatives without having to jump at the obvious solution, which might not always be the best one.
  - ❖ **Processing logic:** discuss the processing logic for the client tier, middle tier and the data tier. Where required add process flow diagrams. Add any pre-process conditions and/or post-process conditions. (Refer Q9 in Java section for design by contract).
  - ❖ **UML diagrams** to communicate the design to the fellow developers, solution designers, architects etc. Usually class diagrams and sequence diagrams are required. The other diagrams may be added for any special cases like (Refer Q107 in Enterprise section):
    - ❖ **State chart diagram:** useful to describe behaviour of an object across several usecases.
    - ❖ **Activity diagram:** useful to express complex operations. Supports and encourages parallel behaviour. Activity and statechart diagrams are beneficial for workflow modelling with multi threaded programming.
    - ❖ **Collaboration and Sequence diagrams:** Use a collaboration or sequence diagram when you want to look at behaviour of several objects within a single use case. If you want to look at a single object across multiple use cases then use statechart.
    - ❖ **Object diagrams:** The Object diagrams show instances instead of classes. They are useful for explaining some complicated objects in detail such as highlighting recursive relationships etc.
    - ❖ **List the package names, class names, database names and table names** with a brief description of their responsibility in a tabular form.
- Prepare a **coding standards** document for the whole team to promote consistency and efficiency. Some coding practices can degrade performance for example:
  - ❖ Inappropriate use of String class. Use StringBuffer instead of String for compute intensive mutations (Refer Q17 in Java section).

- ❖ Code in terms of interface. For example you might decide the `LinkedList` is the best choice for some application, but then later decide `ArrayList` might be a better choice. (Refer **Q15** in Java section)

**Wrong approach** → `List list = new ArrayList();`  
**Right approach** → `List list = new ArrayList(100)`

- ❖ Set the initial capacity of a collection appropriately (e.g. `ArrayList`, `HashMap` etc). (Refer **Q15** in Java section).
- ❖ To promote consistency define standards for variable names, method names, use of logging, curly bracket positions etc.
- Prepare a **code review** document and templates for the whole team. Let us look at some of the elements the code review should cover:
  - ❖ **Proper variable declaration:** e.g. instance versus static variables, constants etc.
  - ❖ **Performance issues:** e.g. Use `ArrayList`, `HashMap` etc instead of `Vector`, `Hashtable` when there is no thread-safety issue.
  - ❖ **Memory issues:** e.g. Improper instantiation of objects instead of object reuse and object pooling, not closing valuable resource in a finally block etc.
  - ❖ **Thread-safety issues:** e.g. Java API classes like `SimpleDateFormat`, `Calendar`, `DecimalFormat` etc are not thread safe, declaring variables in JSP is not thread safe, storing state information in Struts action class or multi-threaded servlet is not thread safe.
  - ❖ **Error handling:** e.g. Re-throwing exception without nesting original exception, EJB methods not throwing EJB exception for system exceptions, etc.
  - ❖ **Use of coding standards:** e.g. not using frameworks, `System.out` is used instead of `log4j` etc.
  - ❖ **Design issues:** No re-use of code, no clear separation of responsibility, invalid use of inheritance to get method reuse, servlets performing JDBC direct access instead of using DAO (Data Access Objects) classes, HTML code in Struts action or servlet classes, servlets used as utility classes rather than as a flow controller etc.
  - ❖ **Documentation of code:** e.g. No comments, no header files etc
  - ❖ **Bugs:** e.g. Calling `setAutoCommit` within container-managed transaction, binary OR “|” used instead of logical OR “||”, relying on pass-by-reference in EJB remote calls, `ResultSet` not being closed on exceptions, EJB methods not throwing `EJBException` for system exceptions etc (Refer **Q76 & Q77** in Enterprise section)
- Prepare additional optional **guideline documents** as per requirements to be shared by the team. This will promote consistency and standards. For example:
  - ❖ Guidelines to setting up J2EE development environment.
  - ❖ Guidelines to version control system (CVS, VSS etc).
  - ❖ Guidelines to deployment steps, environment settings, ant targets etc.
  - ❖ Guidelines for the data modelling (any company standards).
  - ❖ Guidelines for error handling (Refer **Q34, Q35** in Java section & **Q76, Q77** in Enterprise section).
  - ❖ Guidelines for user interface design.
  - ❖ Project overview document.
  - ❖ Software development process document etc.

Some of the above mentioned documents, which are shared by the whole team, can be published in an internal website like Wiki. Wiki is a piece of server software that allows users to freely create and edit Web page content using any Web browser.

**A 02:** Design should be specific to a problem but also should be general enough to address future requirements. Designing reusable object oriented software involves decomposing the business use cases into relevant objects and converting objects into classes.

- Create a **tiered architecture**: client tier, business tier and data tier. Each tier can be further logically divided into layers (Refer Q2, Q3 on Enterprise section). Use MVC (**Model View Controller** architecture for the J2EE and Java based GUI applications).
- Create a **data model**: A data model is a detailed specification of data oriented structures. This is different from the class modelling because it focuses solely on data whereas class models allow you to define both data and behaviour. **Conceptual data models** (aka domain models) are used to explore domain concepts with project stakeholders. **Logical data models** are used to explore the domain concepts, and their relationships. Logical data models depict entity types, data attributes and entity relationships (with Entity Relationship (ER) diagrams). **Physical data models** are used to design the internal schema of a database depicting the tables, columns, and the relationships between the tables. Data models can be created by performing the following tasks:
  - ❖ **Identify entity types, attributes and relationships**: use entity relationship (E-R) diagrams.
  - ❖ **Apply naming conventions (e.g. for tables, attributes, indices, constraints etc)**: Your organization should have standards and guidelines applicable to data modelling.
  - ❖ **Assign keys**: surrogate keys (e.g. assigned by the database like Oracle sequences etc, max() + 1, universally unique identifiers UUIDs, etc), natural keys (e.g. Tax File Numbers, Social Security Numbers etc), and composite keys.
  - ❖ **Normalize to reduce data redundancy and denormalize to improve performance**: Normalized data have the advantage of information being stored in one place only, reducing the possibility of inconsistent data. Furthermore, highly normalized data are loosely coupled. But normalization comes at a performance cost because to determine a piece of information you have to join multiple tables whereas in a denormalized approach the same piece of information can be retrieved from a single row of a table. Denormalization should be used only when performance testing shows that you need to improve database access time for some of your tables.

**Note:** Creating a data model (logical, physical etc) before design model is a matter of preference, but many OO methodologies are based on creating the data model from the object **design model** (i.e. you may need to do some work to create an explicit data model but only after you have a complete OO domain and design model). In many cases when using ORM tools like Hibernate, you do not create the data model at all.

- Create a **design model**: A design model is a detailed specification of the objects and relationships between the objects as well as their behaviour. (Refer Q107 on Enterprise section)
  - ❖ **Class diagram**: contains the implementation view of the entities in the design model. The design model also contains core business classes and non-core business classes like persistent storage, security management, utility classes etc. The class diagrams also describe the structural relationships between the objects.
  - ❖ **Use case realizations**: are described in sequence and collaboration diagrams.
- **Design considerations when decomposing business use cases into relevant classes**: designing reusable and flexible design models requires the following considerations:
  - ❖ **Granularity of the objects** (fine-grained, coarse-grained etc): Can we minimise the network trip by passing a coarse-grained value object instead of making 4 network trips with fine-grained parameters? (Refer Q85 in Enterprise section). Should we use method level (coarse-grained) or code level (fine-grained) thread synchronization? (Refer Q40 in Java section). Should we use a page level access security or a fine-grained programmatic security?
  - ❖ **Coupling between objects** (loosely coupled versus strongly coupled). Should we use business delegate pattern to loosely couple client and business tier? (Refer Q83 in Enterprise section) or Should we use dependency injection? (Refer Q09 in Emerging Technologies/Frameworks).
  - ❖ **Network overheads** for remote objects like EJB, RMI etc: Should we use the session façade, value object patterns? (Refer Q84 & Q85 in Enterprise section).

- ❖ **Definition of class interfaces and inheritance hierarchy:** Should we use an abstract class or an interface? Is there any common functionality that we can move to the super class (or parent class)? Should we use interface inheritance with object composition for code reuse as opposed to implementation inheritance? Etc. (Refer **Q8, Q10** in Java section).
- ❖ **Establishing key relationships** (aggregation, composition, association etc): Should we use aggregation or composition? [composition may require cascade delete] (Refer **Q107, Q108** in Enterprise section – under class diagrams). Should we use an “**is a**” (generalization) relationship or a “**has a**” (composition) relationship? (Refer **Q7** in Java section).
- ❖ **Applying polymorphism and encapsulation:** Should we hide the member variables to improve integrity and security? (Refer **Q8** in Java section). Can we get a polymorphic behaviour so that we can easily add new classes in the future? (Refer **Q8** in Java section).
- ❖ **Applying well-proven design patterns** (like Gang of four design patterns, J2EE design patterns, EJB design patterns etc) help designers to base new designs on prior experience. Design patterns also help you to choose design alternatives (Refer **Q11** in How would you go about...).
- ❖ **Scalability** of the system: **Vertical scaling** is achieved by increasing the number of servers running on a single machine. **Horizontal scaling** is achieved by increasing the number of machines in the cluster. Horizontal scaling is more reliable than the vertical scaling because there are multiple machines involved in the cluster. In vertical scaling the number of server instances that can be run on one machine are determined by the CPU usage and the JVM heap memory.
- ❖ **How do we replicate the session state?** Should we use stateful session beans or HTTP session? Should we serialize this object so that it can be replicated?
- ❖ **Internationalization** requirements for multi-language support: Should we support other languages? Should we support multi-byte characters in the database?
- **Vertical slicing:** Getting the reusable and flexible design the first time is impossible. By developing the initial **vertical slice** of your design you eliminate any nasty integration issues later in your project. Also get the design patterns right early on by building the vertical slice. It will give you experience with what does work and what does not work with Java/J2EE. Once you are happy with the initial vertical slice then you can apply it across the application. The initial vertical slice should be based on a typical business use case. Refer **Q136** in Enterprise section.
- **Ensure the system is configurable** through property files, xml descriptor files, annotations etc. This will improve flexibility and maintainability. Avoid hard coding any values. Use a constant class for values, which rarely change and use property files, xml descriptor files, annotations etc for values, which can change more frequently (e.g. process flow steps etc) and/or environment related configurations(e.g. server name, server port, LDAP server location etc).
- **Design considerations during design, development and deployment phases:** designing a fast, secured, reliable, robust, reusable and flexible system require considerations in the following key areas:
  - ❖ **Performance issues** (network overheads, quality of the code etc): Can I make a single coarse-grained network call to my remote object instead of 3 fine-grained calls?
  - ❖ **Concurrency issues (multi-threading etc)**: What if two threads access my object simultaneously will it corrupt the state of my object?
  - ❖ **Transactional issues (ACID properties)**: What if two clients access the same data simultaneously? What if one part of the transaction fails, do we rollback the whole transaction? What if the client resubmits the same transactional page again?
  - ❖ **Security issues**: Are there any potential security holes for SQL injection or URL injection by hackers?
  - ❖ **Memory issues**: Is there any potential memory leak problems? Have we allocated enough heap size for the JVM? Have we got enough perm space allocated since we are using 3<sup>rd</sup> party libraries, which generate classes dynamically? (e.g. JAXB, XSLT, JasperReports etc)
  - ❖ **Scalability issues**: Will this application scale vertically and horizontally if the load increases? Should this object be serializable? Does this object get stored in the HttpSession?

- ❖ **Maintainability, reuse, extensibility etc:** How can we make the software reusable, maintainable and extensible? What design patterns can we use? How often do we have to refactor our code?
  - ❖ **Logging and auditing** if something goes wrong can we look at the logs to determine the root cause of the problem?
  - ❖ **Object life cycles:** Can the objects within the server be created, destroyed, activated or passivated depending on the memory usage on the server? (e.g. EJB).
  - ❖ **Resource pooling:** Creating and destroying valuable resources like database connections, threads etc can be expensive. So if a client is not using a resource can it be returned to a pool to be reused when other clients connect? What is the optimum pool size?
  - ❖ **Caching** can we save network trips by storing the data in the server's memory? How often do we have to clear the cache to prevent the in memory data from becoming stale?
  - ❖ **Load balancing:** Can we redirect the users to a server with the lightest load if the other server is overloaded?
  - ❖ **Transparent fail over:** If one server crashes can the clients be routed to another server without any interruptions?
  - ❖ **Clustering:** What if the server maintains a state when it crashes? Is this state replicated across the other servers?
  - ❖ **Back-end integration:** How do we connect to the databases and/or legacy systems?
  - ❖ **Clean shutdown:** Can we shut down the server without affecting the clients who are currently using the system?
  - ❖ **Systems management:** In the event of a catastrophic system failure who is monitoring the system? Any alerts or alarms? Should we use JMX? Should we use any performance monitoring tools like Tivoli etc?
  - ❖ **Dynamic redeployment:** How do we perform the software deployment while the site is running? (Mainly for mission critical applications 24hrs X 7days).
  - ❖ **Portability issues:** Can I port this application to a different server 2 years from now?
- 

**Q 03:** How would you go about identifying performance and/or memory issues in your Java/J2EE application?

**A 03:** Profiling can be used to identify any performance issues or memory leaks. Profiling can identify what lines of code the program is spending the most time in? What call or invocation paths are used to reach at these lines? What kinds of objects are sitting in the heap? Where is the memory leak? Etc.

- There are many tools available for the optimization of Java code like **JProfiler**, **Borland Optimizelt** etc. These tools are very powerful and easy to use. They also produce various reports with graphs.

Optimizeit™ Request Analyzer provides advanced profiling techniques that allow developers to analyse the performance behaviour of code across J2EE application tiers. Developers can efficiently prioritize the performance of Web requests, JDBC, JMS, JNDI, JSP, RMI, and EJB so that trouble spots can be proactively isolated earlier in the development lifecycle.

**Thread Debugger tools** can be used to identify threading issues like thread starvation and contention issues that can lead to system crash.

**Code coverage tools** can assist developers with identifying and removing any dead code from the applications.

- **Hprof** which comes with JDK for free. Simple tool.

```
Java -Xprof myClass
```

```
java -Xrunhprof:[help][[<option>=<value>]
java -Xrunhprof:cpu=samples, depth=6, heap=sites
```

- Use operating system process monitors like NT/XP Task Manager on PCs and commands like ps, iostat, netstat, vmstat, uptime, nfsstat etc on UNIX machines.
- Write your own wrapper MemoryLogger and/or PerformanceLogger utility classes with the help of **totalMemory()** and **freeMemory()** methods in the Java **Runtime** class for memory usage and **System.currentTimeMillis()** method for performance. You can place these MemoryLogger and PerformanceLogger calls strategically in your code. Even better approach than utility classes is using Aspect Oriented Programming (AOP) for pre and post memory and/or performance recording where you have the control of activating memory/performance measurement only when needed. (Refer **Q3 – Q5** in Emerging Technologies/Frameworks section).

**Q 04:** How would you go about minimising memory leaks in your Java/J2EE application?

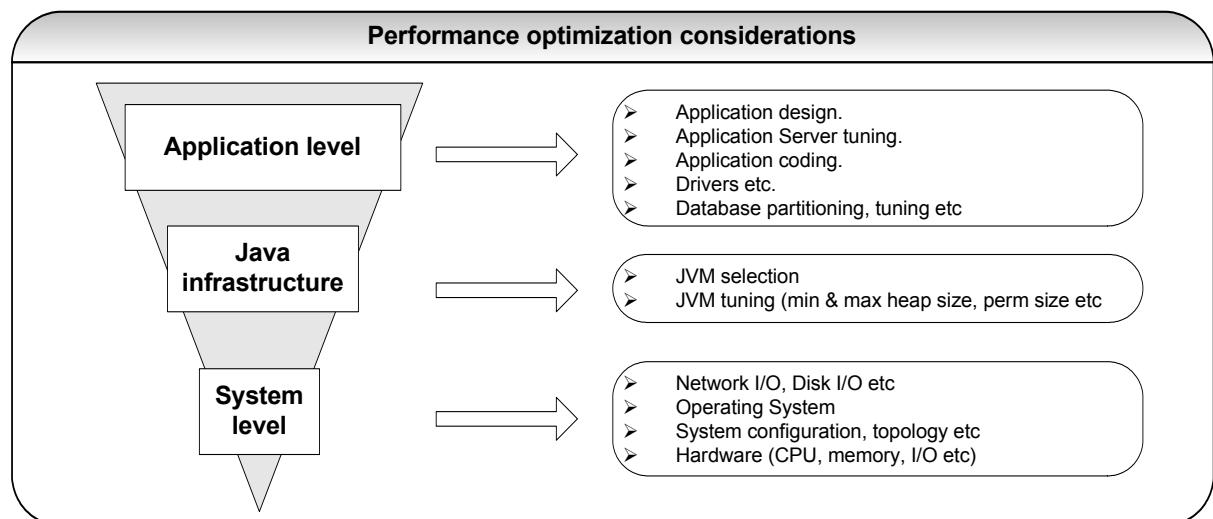
**A 04:** Java's memory management (i.e. Garbage Collection) prevents lost references and dangling references but it is still possible to create memory leaks in other ways. If the application runs with memory leaks for a long duration you will get the error **java.lang.OutOfMemoryError**.

In Java, typically the memory leak occurs when **an object of a longer lifecycle has a reference to the objects of a short life cycle**. This prevents the objects with short life cycle being garbage collected. The developer must remember to remove the reference to the short-lived objects from the long-lived objects. Objects with the same life cycle do not cause any problem because the garbage collector is smart enough to deal with the circular references (Refer **Q33** in Java section).

- Java collection classes like Hashtable, ArrayList etc maintain references to other objects. So having a long life cycle ArrayList pointing to many short-life cycle objects can cause memory leaks.
- Commonly used **singleton design pattern** (Refer **Q45** in Java section) can cause memory leaks. Singletons typically have a long life cycle. If a singleton has an ArrayList or a Hashtable then there is a potential for memory leaks.
- Java programming language includes a finalize method that allows an object to free system resources, in other words, to clean up after itself. However using finalize doesn't guarantee that a class will clean up resources expediently. A better approach for cleaning up resources involves the finally method and an explicit close statement. So freeing up the valuable resource in the finalize method or try {} block instead of finally {} block can cause memory leaks (Refer **Q45** in Enterprise section).

**Q 05:** How would you go about improving performance in your Java/J2EE application?

**A 05:** The performance bottlenecks can be attributed to one or more of the following:



Let us discuss some of the aspects in detail:

- Java/J2EE application code related performance bottlenecks:
  - ❖ Refer **Q63** in Java section.

- ❖ Refer **Q123, Q125** in Enterprise section.
- Java/J2EE design related performance bottlenecks. Application design is one of the most important considerations for performance. A well-designed application will not only avoid many performance pitfalls but will also be easier to maintain and modify during the performance-testing phase of the project.
  - ❖ Use proper design patterns to **minimise network trips** (session facade, value object Refer etc **Q83-Q87** in Enterprise section).
  - ❖ **Minimise serialization cost** by implementing session beans with remote interfaces and entity beans with local interfaces (applicable to EJB 2.x) or even the session beans can be implemented with local interfaces sharing the same JVM with the Web tier components. For EJB1.x some EJB containers can be configured to use **pass-by-reference** instead of pass-by-value (pass-by-value requires serialization) Refer **Q69, Q82** in Enterprise section.
  - ❖ Use of multi-threading from a thread-pool (say 10 – 50 threads). Using a large number of threads adversely affects performance by consuming memory through thread stacks and CPU by context switching.
- Database related performance bottlenecks.
  - ❖ Use proper database indexes. Numeric indices are more efficient than character based indices. Minimise the number of columns in your composite keys. Performing a number of “INSERT” operations is more efficient when fewer columns are indexed and “SELECT” operations are more efficient when, adequately indexed based on columns frequently used in your “WHERE” clause. So it is a trade-off between “SELECT” and “INSERT” operations.
  - ❖ Minimise use of composite keys or use fewer columns in your composite keys.
  - ❖ Partition the database for performance based on the most frequently accessed data and least frequently accessed data.
  - ❖ Identify and optimise your SQL queries causing performance problems (Refer **Q97** in Enterprise section).
  - ❖ De-normalise your tables where necessary for performance (Refer **Q98** in Enterprise section).
  - ❖ Close database connections in your Java code in the finally block to avoid any ‘open cursors’ problem (Refer **Q45** in Enterprise section).
  - ❖ Use **optimistic concurrency** as opposed to **pessimistic concurrency** where appropriate (Refer **Q78** in Enterprise section).
- Application Server, JVM, Operating System, and/or hardware related performance bottlenecks.
  - ❖ **Application Server:** Configure the application server for optimum performance (Refer **Q88, Q123** in Enterprise section).
  - ❖ **Operating System:** Check for any other processes clogging up the system resources, maximum number of processes it can support or connect, optimise operating system etc.
  - ❖ **Hardware:** Insufficient memory, insufficient CPU, insufficient I/O, limitation of hardware configurations, network constraints like bandwidth, message rates etc.

**Q 06:** How would you go about identifying any potential thread-safety issues in your Java/J2EE application?

**A 06:** When you are writing graphical programs like Swing or Internet programs using servlets or JSPs multi-threading is a necessity for all but some special and/or trivial programs.

An application program or a process can have multiple threads like multiple processes that can run on one computer. The multiple threads appear to be doing their work in parallel. When implemented on a multi-processor machine, they can actually work in parallel.

Unlike processes, threads share the same address space (Refer **Q36** in Java section) which means they can read and write the same variables and data structures. So care should be taken to avoid one thread disturbing the work of another thread. Let us look at some of the common situations where care should be taken:

- ❖ Swing components can only be accessed by one thread at a time. A few operations are guaranteed to be thread safe but the most others are not. Generally the Swing components should be accessed through an **event-dispatching thread**. (Refer **Q53** in Java section).
- ❖ A typical Servlet life cycle creates a single instance of each servlet and creates multiple threads to handle the service() method. **The multi-threading aids efficiency but the servlet code must be coded in a thread safe manner.** The shared resources (e.g. instance variable) should be appropriately synchronized or should only use variables in a read-only manner. (Refer **Q16** in Enterprise section).
- ❖ The declaration of variables in JSP is not thread-safe, because the declared variables end up in the generated servlet as an instance variable, not within the body of the \_jspService() method. (Refer **Q34** in Enterprise section).
- ❖ Struts framework action classes are not thread-safe. (Refer **Q113** in Enterprise section).
- ❖ Some Java collection classes like Hashmap, ArrayList etc are not thread-safe. (Refer **Q13** in Java section).
- ❖ Some of the Java core library classes are not thread safe. For e.g. java.util.SimpleDateFormat, java.util.Locale etc.

**Q 07:** How would you go about identifying any potential transactional issues in your Java/J2EE application?

**A 07:**

- ❖ When a connection is created, it is in **auto-commit** mode. This means that each individual SQL statement is treated as a transaction and will be automatically committed immediately after it is executed. The way to allow two or more statements to be grouped into a transaction is to **disable** auto-commit mode. (Refer **Q43** in Enterprise section). Disabling auto-commit mode can improve performance by minimising number of times it accesses the database.
- ❖ A transaction is often described by ACID properties (Atomic, Consistent, Isolated and Durable). A **distributed transaction** is an ACID transaction between two or more independent transactional resources like two separate databases. For a transaction to commit successfully, all of the individual resources must commit successfully. If any of them are unsuccessful, the transaction must rollback in all of the resources. A **2-phase commit** is an approach for committing a distributed transaction in 2 phases. Refer **Q73** in Enterprise section.
- ❖ Isolation levels provide a degree of control of the effects one transaction can have on another concurrent transaction. Concurrent effects are determined by the precise ways in which, a particular relational database handles locks and its drivers may handle these locks differently. Isolation levels are used to overcome transactional problems like lost update, uncommitted data (aka dirty reads), inconsistent data (aka. phantom update), and phantom insert. Higher isolation levels can adversely affect performance at the expense of data accuracy. Refer **Q72** in Enterprise section.

Isolation Level	Lost Update	Uncommitted Data	Inconsistent Data	Phantom Insert
Read Uncommitted	Prevented by DBMS	Can happen	Can happen	Can happen
Read Committed	Prevented by DBMS	Prevented by DBMS	Can happen	Can happen
Repeatable Read	Prevented by DBMS	Prevented by DBMS	Prevented by DBMS	Can happen
Serializable	Prevented by DBMS	Prevented by DBMS	Prevented by DBMS	Prevented by DBMS

- ❖ Decide between optimistic and pessimistic concurrency control. (Refer **Q78** in Enterprise section).
- ❖ Evaluate a strategy to determine if the data is stale when using strategies to cache data. (Refer **Q79** in Enterprise section).

#### EJB related transactional issues:

- ❖ Set the appropriate transactional attributes for the EJBs. (Refer **Q71** in Enterprise section).
- ❖ Set the appropriate isolation level for the EJB. The isolation level should not be any more restrictive than it has to be. Higher isolation levels can adversely affect performance. (Refer **Q72** in Enterprise section). Isolation levels are application server specific and not part of the standard EJB configuration.

- ❖ In EJB 2.x, transactions are rolled back by the container when a **system exception** is thrown. When an **application exception** is thrown then the transactions are not rolled back by the container. So the developer has to roll it back using `ctx.setRollbackOnly()` call. (Refer Q76, Q77 in Enterprise section).
- ❖ Detect doomed transactions to avoid performing any unnecessary compute intensive operations. (Refer Q72 in Enterprise section).

**Q 08:** How would you go about applying the Object Oriented (OO) design concepts in your Java/J2EE application?

**A 08:**

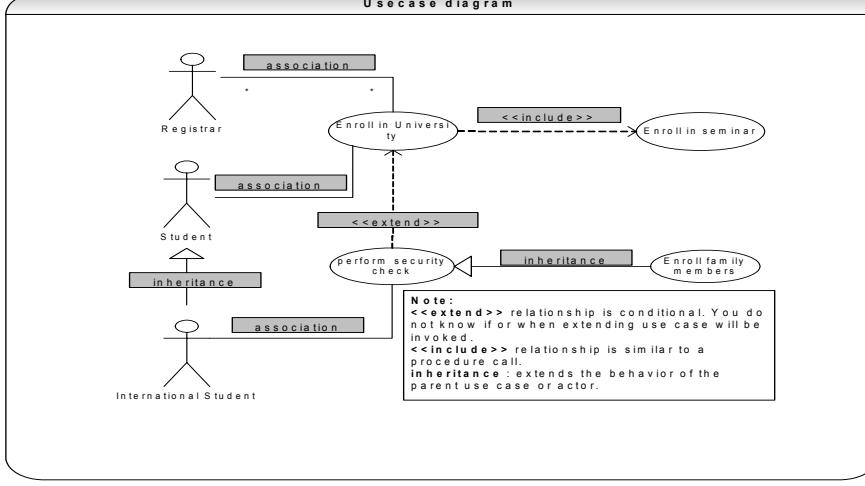
Question	Answer
What are the key characteristics of an OO language like Java?	<p>A true object oriented language should support the following 3 characteristics:</p> <ul style="list-style-type: none"> <li>❖ <b>Encapsulation</b> (aka <b>information hiding</b>): implements information hiding and modularity (abstraction).</li> <li>❖ <b>Polymorphism</b>: The same message sent to different objects, results in behaviour that is dependent on the nature of the object receiving the message.</li> <li>❖ <b>Inheritance</b>: Encourages code reuse and code organisation by defining the new class based on the existing class.</li> </ul> <p><b>What is dynamic binding?</b></p> <p><b>Dynamic binding (aka late binding)</b>: The dynamic binding is used to implement polymorphism. Objects could come from local process or from across the network from a remote process. We should be able to send messages to objects without having to know their types at the time of writing the code. Dynamic binding provides maximum flexibility at the execution time. Usually dynamic binding or late binding takes a small performance hit.</p> <p>Refer Q8 in Java section.</p> <p>Let us take an example to illustrate dynamic binding through polymorphic behaviour:</p> <p>Say you have a method in Java</p> <pre>void draw(Shape s) {     s.erase();     // ...     s.draw(); }</pre> <p>The above method will talk to any shape, so it is independent of the specific type of object it is erasing and drawing. Now let us look at some other program, which is making use of this <code>draw(Shape s)</code> method:</p> <pre>Circle cir = new Circle(); Square sq = new Square();  draw(cir); draw(sq);</pre> <p>So the interesting thing is that the method call to <code>draw(Shape s)</code> will cause different code to be executed. So you send a message to an object even though you don't know what specific type it is and the right thing happens. This is called dynamic binding, which gives you polymorphic behaviour.</p>
How will you decide whether to use an interface or an abstract class?	<ul style="list-style-type: none"> <li>❖ <b>Abstract Class</b>: Often in a design, you want the base class to present <b>only an interface</b> for its derived classes. That is, you don't want anyone to actually create an object of the base class, only to upcast to it so that its interface can be used. This is accomplished by making that class <b>abstract</b> using the <b>abstract</b> key word. If anyone tries to make an object of an <b>abstract</b> class, the compiler prevents them. This is a tool to enforce a particular design.</li> <li>❖ <b>Interface</b>: The <b>interface</b> key word takes the concept of an <b>abstract</b> class one step further by preventing any function definitions at all. An <b>interface</b> is a very useful and commonly used tool, as it provides the perfect separation of interface and implementation. In addition, you can combine many interfaces together, if you wish. (You cannot inherit from more than one regular <b>class</b> or <b>abstract class</b>.)</li> </ul> <p>Now the design decision...</p>

	<p><b>When to use an Abstract Class:</b> Abstract classes are excellent candidates inside of application frameworks. Abstract classes let you define some default behaviour and force subclasses to provide any specific behaviour.</p> <p><b>When to use an Interface:</b> If you need to change your design frequently, I prefer using interface to abstract. For example, the strategy pattern lets you swap new algorithms and processes into your program without altering the objects that use them. <b>Example: Strategy Design Pattern.</b></p> <p>Another justification of interfaces is that they solved the '<b>diamond problem</b>' of traditional multiple inheritance. Java does not support multiple inheritances. Java only supports <b>multiple interface inheritance</b>. Interface will solve all the ambiguities caused by this 'diamond problem'. Refer <b>Q10</b> in Java section.</p> <p><b>Interface inheritance vs. Implementation inheritance:</b> Prefer interface inheritance to implementation inheritance because it promotes the design concept of <b>coding to an interface</b> and <b>reduces coupling</b>. Interface inheritance can achieve <b>code reuse</b> with the help of <b>object composition</b>. Refer <b>Q08</b> in Java section.</p>								
Why abstraction is important in Object Oriented programming?	<p>The software you develop should optimally cater for the current requirements and problems and also should be flexible enough to easily handle future changes.</p> <p>Abstraction is an important OO concept. The ability for a program to ignore some aspects of the information that it is manipulating, ie. Ability to focus on the essential. Each object in the system serves as a model of an abstract "actor" that can perform work, report on and change its state, and "communicate" with other objects in the system, without revealing <i>how</i> these features are implemented. Abstraction is the process where ideas are distanced from the concrete implementation of the objects. <b>The concrete implementation will change but the abstract layer will remain the same.</b></p> <p><b>Let us look at an analogy:</b></p> <p>When you drive your car you do not have to be concerned with the exact internal working of your car (unless you are a mechanic). What you are concerned with is interacting with your car via its interfaces like steering wheel, brake pedal, accelerator pedal etc. Over the years a car's engine has improved a lot but its basic interface has not changed (ie you still use steering wheel, brake pedal, accelerator pedal etc to interact with your car). This means that the <b>implementation</b> has changed over the years but the <b>interface</b> remains the same. Hence the knowledge you have of your car is abstract.</p>								
Explain black-box reuse and white-box reuse? Should you favour Inheritance (white-box reuse) or aggregation (black-box reuse)?	<p><b>Black-box reuse</b> is when a class uses another class without knowing the internal contents of it. The black-box reuses are:</p> <ul style="list-style-type: none"> <li>❖ <b>Dependency</b> is the weakest type of black-box reuse.</li> <li>❖ <b>Association</b> is when one object knows about or has a relationship with the other objects.</li> <li>❖ <b>Aggregation</b> is the <b>whole part relationship</b> where one object contains one or more of the other objects.</li> <li>❖ <b>Composition</b> is a stronger <b>whole part relationship</b></li> </ul> <p>Refer <b>Q107, Q108</b> in Enterprise section</p> <p><b>White-box reuse</b> is when a class knows internal contents of another class. E.g. <b>inheritance</b> is used to modify implementation for reusability.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #cccccc;"> <th style="text-align: left; padding: 2px;"><b>Aggregation (Black-box reuse)</b></th> <th style="text-align: left; padding: 2px;"><b>Inheritance (White-box reuse)</b></th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">Defined dynamically or at run time via object references. Since only interfaces are used, it has the advantage of maintaining the integrity (ie encapsulation).</td> <td style="padding: 2px;">Inheritance is defined statically or at compile time. Inheritance allows an easy way to modify implementation for reusability.</td> </tr> <tr> <td style="padding: 2px;">Disadvantage of aggregation is that it increases the number of objects and relationships.</td> <td style="padding: 2px;">A disadvantage of inheritance is that it breaks encapsulation, which implies implementation dependency. This means when you want to carry out the redesign the super class (ie parent class) has to be modified or replaced which is more likely to affect the subclasses as well. In general it will affect the whole inheritance hierarchy.</td> </tr> <tr> <td></td> <td style="text-align: right; padding: 2px;"><b>Verdict:</b> So the tendency is to favour aggregation over inheritance.</td> </tr> </tbody> </table>	<b>Aggregation (Black-box reuse)</b>	<b>Inheritance (White-box reuse)</b>	Defined dynamically or at run time via object references. Since only interfaces are used, it has the advantage of maintaining the integrity (ie encapsulation).	Inheritance is defined statically or at compile time. Inheritance allows an easy way to modify implementation for reusability.	Disadvantage of aggregation is that it increases the number of objects and relationships.	A disadvantage of inheritance is that it breaks encapsulation, which implies implementation dependency. This means when you want to carry out the redesign the super class (ie parent class) has to be modified or replaced which is more likely to affect the subclasses as well. In general it will affect the whole inheritance hierarchy.		<b>Verdict:</b> So the tendency is to favour aggregation over inheritance.
<b>Aggregation (Black-box reuse)</b>	<b>Inheritance (White-box reuse)</b>								
Defined dynamically or at run time via object references. Since only interfaces are used, it has the advantage of maintaining the integrity (ie encapsulation).	Inheritance is defined statically or at compile time. Inheritance allows an easy way to modify implementation for reusability.								
Disadvantage of aggregation is that it increases the number of objects and relationships.	A disadvantage of inheritance is that it breaks encapsulation, which implies implementation dependency. This means when you want to carry out the redesign the super class (ie parent class) has to be modified or replaced which is more likely to affect the subclasses as well. In general it will affect the whole inheritance hierarchy.								
	<b>Verdict:</b> So the tendency is to favour aggregation over inheritance.								

<p>What is your understanding on Aspect Oriented Programming (AOP)?</p>	<p>Aspect-Oriented Programming (<b>AOP</b>) <u>complements</u> <b>OO</b> programming by allowing developers to dynamically modify the static OO model to create a system that can grow to meet new requirements.</p> <p>AOP allows us to dynamically modify our static model to include the code required to fulfil the secondary requirements (like auditing, logging, security, exception handling etc) without having to modify the original static model (in fact, we don't even need to have the original code). Better still, we can often keep this additional code in a single location rather than having to scatter it across the existing model, as we would have to if we were using OO on its own. (Refer <b>Q3 –Q5</b> in Emerging Technologies/Frameworks section.)</p> <p><b>For example</b> A typical Web application will require a servlet to bind the HTTP request to an object and then passes to the business handler object to be processed and finally return the response back to the user. So initially only a minimum amount of code is required. But once you start adding all the other additional secondary requirements (aka <b>crosscutting concerns</b>) like logging, auditing, security, exception-handling etc the code will inflate to 2-4 times its original size. This is where AOP can help.</p>
-------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Q 09:** How would you go about applying the UML diagrams in your Java/J2EE project?

**A 09:**

Question	Answer
<p>Explain the key relationships in the use case diagrams?</p>	<p>Refer <b>Q107</b> in Enterprise section. Use case has 4 types of relationships:</p> <p><b>Between actor and use case</b></p> <ul style="list-style-type: none"> <li>❖ <b>Association:</b> Between <b>actor</b> and <b>use case</b>. May be navigable in both directions according to the initiator of the communication between the actor and the usecase.</li> </ul> <p><b>Between use cases</b></p> <ul style="list-style-type: none"> <li>❖ <b>Extends:</b> This is an optional extended behaviour of a use case. This behaviour is executed only under certain conditions such as performing a security check etc.</li> <li>❖ <b>Includes:</b> This specifies that the base use case needs an additional use case to fully describe its process. It is mainly used to show common functionality that is shared by several use cases.</li> <li>❖ <b>Inheritance (or generalization):</b> Child use case inherits the behaviour of its parent. The child may override or add to the behaviour of the parent.</li> </ul>  <p><b>Note:</b>  <b>&lt;&lt;extend&gt;&gt;</b> relationship is conditional. You do not know if or when extending use case will be triggered.  <b>&lt;&lt;include&gt;&gt;</b> relationship is similar to a procedure call.  <b>Inheritance</b> : extends the behavior of the parent use case or actor.</p>
<p>What is the main difference between the collaboration diagram and the sequence diagram?</p>	<p>Refer <b>Q107</b> in Enterprise section:</p> <p>Collaboration diagrams convey the same message as sequence diagrams but the collaboration diagrams focus on <b>object roles</b> instead of <b>times in which</b> the messages are sent. The sequence diagram is time line driven.</p>
<p>When to use various UML diagrams?</p>	<p>Refer <b>Q107</b> in Enterprise section.</p> <ul style="list-style-type: none"> <li>❖ <b>Use case diagrams:</b> <ul style="list-style-type: none"> <li>▪ Determining the user requirements. New use cases often generate new requirements.</li> <li>▪ Communicating with clients. The simplicity of the diagram makes use case diagrams a good way for designers and developers to communicate with clients.</li> <li>▪ Generating test cases. Each scenario for the use case may suggest a suite of test</li> </ul> </li> </ul>

	<p>cases.</p> <ul style="list-style-type: none"> <li>❖ <b>Class diagrams:</b> <ul style="list-style-type: none"> <li>▪ Class diagrams are the backbone of <b>Object Oriented</b> methods. So they are used frequently.</li> <li>▪ Class diagrams can have a conceptual perspective and an implementation perspective. During the analysis draw the conceptual model and during implementation draw the implementation model.</li> </ul> </li> <li>❖ <b>Interaction diagrams (Sequence and/or Collaboration diagrams):</b> <ul style="list-style-type: none"> <li>▪ When you want to look at behaviour of <b>several objects within a single use case</b>. If you want to look at a <b>single object across multiple use cases</b> then use statechart diagram as described below.</li> </ul> </li> <li>❖ <b>State chart diagrams:</b> <ul style="list-style-type: none"> <li>▪ Statechart diagrams are good at describing the <b>behaviour of an object across several use cases</b>. But they are not good at describing the interaction or collaboration between many objects. Use interaction and/or activity diagrams in conjunction with the statechart diagram to communicate complex operations involving multi-threaded programs etc.</li> <li>▪ Use it only for classes that have complex state changes and behaviour. <b>For example:</b> the User Interface (UI) control objects, Objects shared by multi-threaded programs etc.</li> </ul> </li> <li>❖ <b>Activity diagram:</b> <ul style="list-style-type: none"> <li>▪ <b>Activity</b> and <b>Statechart</b> diagrams are generally useful to express complex operations. The great strength of activity diagrams is that they support and encourage parallel behaviour. An activity and statechart diagrams are beneficial for workflow modelling with multi threaded programming.</li> </ul> </li> </ul>
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Q 10:** How would you go about describing the software development processes you are familiar with?

**A 10:** In addition to technical questions one should also have a good understanding of the software development process.

Question	Answer
What is the key difference between the waterfall approach and the iterative approach to software development? How to decide which one to use?	<p>Refer Q103 – Q105 in Enterprise section</p> <p><b>Waterfall</b> approach is sequential in nature. The <b>iterative</b> approach is non-sequential and incremental. The iterative and incremental approach has been developed based on the following:</p> <ul style="list-style-type: none"> <li>• <b>You can't express all your needs up front.</b> It is usually not feasible to define in detail (that is, before starting full-scale development) the operational capabilities and functional characteristics of the entire system. These usually evolve over time as development progresses.</li> <li>• <b>Technology changes over time.</b> Some development lifecycle spans a long period of time during which, given the pace at which technology evolves, significant technological shifts may occur.</li> <li>• <b>Complex systems.</b> This means it is difficult to cope with them adequately unless you have an approach for mastering complexity.</li> </ul> <p><b><u>How to decide which one to use?</u></b></p> <p><b>Waterfall</b> approach is more suitable in the following circumstances:</p> <ul style="list-style-type: none"> <li>• Have a small number of unknowns and risks. That is if <ul style="list-style-type: none"> <li>• It has a known domain.</li> <li>• The team is experienced in current process and technology.</li> <li>• There is no new technology.</li> <li>• There is a pre-existing architecture baseline.</li> </ul> </li> <li>• Is of short duration (two to three months).</li> <li>• Is an evolution of an existing system?</li> </ul> <p>The <b>iterative</b> approach is more suitable (Refer Q136 in Enterprise Section)</p> <ul style="list-style-type: none"> <li>• Have a large number of unknowns and risks. So it pays to design, develop and test a vertical slice iteratively and then replicate it through other iterations. That is if</li> </ul>

	<ul style="list-style-type: none"> <li>Integrating with new systems.</li> <li>New technology and/or architecture.</li> <li>The team is fairly keen to adapt to this new process.</li> </ul> <ul style="list-style-type: none"> <li>Is of large duration (longer than 3 months).</li> <li>Is a new system?</li> </ul>
Have you used extreme programming techniques? Explain?	<p>Extreme Programming (or XP) is a set of values, principles and practices for rapidly developing high-quality software that provides the highest value for the customer in the fastest way possible. <b>XP is a minimal instance of RUP.</b> XP is extreme in the sense that it takes <b>12 well-known software development "best practices" to their logical extremes.</b></p> <p>The 12 core practices of XP are:</p> <ol style="list-style-type: none"> <li><b>The Planning Game:</b> Business and development cooperate to produce the maximum business value as rapidly as possible. The planning game happens at various scales, but the basic rules are always the same: <ul style="list-style-type: none"> <li>Business comes up with a list of desired features for the system. Each feature is written out as a <b>user story</b> (or PowerPoint screen shots with changes highlighted), which gives the feature a name, and describes in broad strokes what is required. User stories are typically written on 4x6 cards.</li> <li>Development team estimates how much effort each story will take, and how much effort the team can produce in a given time interval (i.e. the iteration).</li> <li>Business then decides which stories to implement in what order, as well as when and how often to produce production releases of the system.</li> </ul> </li> <li><b>Small releases:</b> Start with the smallest useful feature set. Release early and often, adding a few features each time.</li> <li><b>System metaphor:</b> Each project has an organising metaphor, which provides an easy to remember naming convention.</li> <li><b>Simple design:</b> Always use the simplest possible design that gets the job done. The requirements will change tomorrow, so only do what's needed to meet today's requirements.</li> <li><b>Continuous testing:</b> Before programmers add a feature, they write a test for it. Tests in XP come in two basic flavours. <ul style="list-style-type: none"> <li><b>Unit tests</b> are automated tests written by the developers to test functionality as they write it. Each unit test typically tests only a single class, or a small cluster of classes. Unit tests are typically written using a unit-testing framework, such as <b>JUnit</b>.</li> <li><b>Customer to test that the overall system is functioning as specified, defines acceptance tests (aka Functional tests).</b> Acceptance tests typically test the entire system, or some large chunk of it. When all the acceptance tests pass for a given user story, that story is considered complete. At the very least, an acceptance test could consist of a script of user interface actions and expected results that a human can run. Ideally acceptance tests should be automated using frameworks like Canoo Web test, Selenium Web test etc.</li> </ul> </li> <li><b>Refactoring:</b> Refactor out any duplicate code generated in a coding session. You can do this with confidence that you didn't break anything because you have the tests.</li> <li><b>Pair Programming:</b> All production code is written by two programmers sitting at one machine. Essentially, all code is reviewed as it is written.</li> <li><b>Collective code ownership:</b> No single person "owns" a module. Any developer is expected to be able to work on any part of codebase at any time.</li> <li><b>Continuous integration:</b> All changes are integrated into codebase at least daily. The tests have to run 100% both before and after integration. You can use tools like Ant, CruiseControl, and Maven etc to continuously build and integrate your code.</li> <li><b>40-Hour Workweek:</b> Programmers go home on time. In crunch mode, up to one week of overtime is allowed. But multiple consecutive weeks of overtime are treated as a sign that something is very wrong with the process.</li> <li><b>On-site customer:</b> Development team has continuous access to a real live customer or business owner, that is, someone who will actually be using the system. For commercial software with lots of customers, a customer proxy (usually the product manager, Business</li> </ol>

	<p>Analyst etc) is used instead.</p> <p>12. <b>Coding standards:</b> Everyone codes to the same standards. Ideally, you shouldn't be able to tell by looking at it, which developer on the team has touched a specific piece of code.</p> <p>A typical extreme programming project will have:</p> <ul style="list-style-type: none"> <li>• All the programmers in a room together usually sitting around a large table.</li> <li>• Fixed number of iterations where <b>each iteration takes 1-3 weeks</b>. At the beginning of each iteration get together with the customer.</li> <li>• Pair-programming.</li> <li>• Writing test cases first (i.e. TDD).</li> <li>• Delivery of a functional system at the end of 1-3 week iteration.</li> </ul>
Have you used agile (i.e. Lightweight) software development methodologies?	<p><b>Agile (i.e. lightweight) software development process</b> is gaining popularity and momentum across organizations. Several methodologies fit under this agile development methodology banner. All these methodologies share many characteristics like <b>iterative and incremental development, test driven development</b> (i.e. TDD), <b>stand up meetings to improve communication, automatic testing, build and continuous integration of code</b> etc. Among all the agile methodologies XP is the one which has got the most attention. Different companies use different flavours of agile methodologies by using different combinations of methodologies (e.g. primarily XP with other methodologies like Scrum, FDD, TDD etc). Refer <b>Q136</b> in Enterprise section.</p>

**Q 11:** How would you go about applying the design patterns in your Java/J2EE application?

**A 11:** It is really worth reading books and articles on design patterns. It is sometimes hard to remember the design patterns, which you do not use regularly. So if you do not know a particular design pattern you can always honestly say that you have not used it and subsequently suggest that you can explain another design pattern, which you have used recently or more often. It is always challenging to decide, which design pattern to use when? How do you improve your design pattern skills? Practice, practice, practice. I have listed some of the design patterns below with scenarios and examples:

**Note:** To keep it simple, System.out.println(...) is used. In real practice, use logging frameworks like log4j. Also package constructs are not shown. In real practice, each class should be stored in their relevant packages like com.items etc. Feel free to try these code samples by typing them into a Java editor of your choice and run the main class *Shopping*. Also constants should be declared in a typesafe manner as shown below:

```
/*
 * use typesafe enum pattern as shown below if you are using below JDK 1.5 or use "enum" if you are using JDK 1.5
 */
public class ItemType {
 private final String name;

 public static final ItemType Book = new ItemType("book");
 public static final ItemType CD = new ItemType("cd");
 public static final ItemType COSMETICS = new ItemType("cosmetics");
 public static final ItemType CD_IMPORTED = new ItemType("cd_imported");

 private ItemType(String name) {this.name = name;}
 public String toString() {return name;}
 //add compareTo(), readResolve() methods etc as required ...
}
```

**Scenario:** A company named XYZ Retail is in the business of selling *Books*, *CDs* and *Cosmetics*. *Books* are sales tax exempt and *CDs* and *Cosmetics* have a sales tax of 10%. *CDs* can be imported and attracts an import tax of 5%. Write a shopping basket program, which will calculate extended price (*qty* \* (*unitprice* + *tax*)) inclusive of tax for each item in the basket, total taxes and grand total.

**Solution:** Sample code for the items (i.e. Goods) sold by XYZ Retail. Let's define an *Item* interface to follow the design principle of **code to an interface not to an implementation**. **CO**

```
public interface Item {
 public static final int TYPE_BOOK = 1;
 public static final int TYPE_CD = 2;
 public static final int TYPE_COSMETICS = 3;
 public static final int TYPE_CD_IMPORTED = 4;

 public double getExtendedTax();
```

```
public double getExtendedTaxPrice() throws ItemException;
public void setImported(boolean b);
public String getDescription();
}
```

The following class **Goods** cannot be instantiated (since it is **abstract**). You use this abstract class to achieve **code reuse**.

```
/*
 * abstract parent class, which promotes code reuse for all the subclasses
 * like Book, CD, and Cosmetics. implements interface Item to
 * promote design principle code to interface not to an implementation.
 */
public abstract class Goods implements Item {
 //define attributes
 private String description;
 private int qty;
 private double price;
 private Tax tax = new Tax();

 public Goods(String description, int qty, double price) {
 this.description = description;
 this.qty = qty;
 this.price = price;
 }

 protected abstract boolean isTaxed();
 protected abstract boolean isImported();

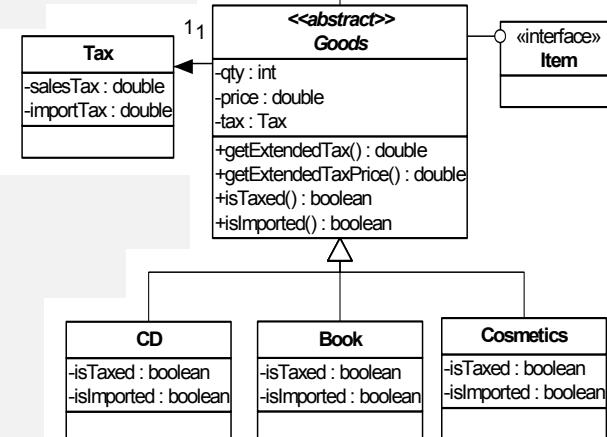
 public double getExtendedTax() {
 tax.calculate(isTaxed(), isImported(), price);
 return this.tax.getTotalUnitTax() * qty;
 }

 public double getExtendedTaxPrice() throws ItemException {
 if (tax == null) {
 throw new ItemException("Tax should be calculated first:");
 }
 return qty * (this.tax.getTotalUnitTax() + price);
 }

 //getters and setters go here for attributes like description etc
 public String getDescription() {
 return description;
 }

 public String toString() {
 return qty + " " + description + " : ";
 }
}
```

code reuse is achieved through implementation inheritance.



The **Book**, **CD** and **Cosmetics** classes can be written as shown below:

```
public class Book extends Goods {
 private boolean isTaxed = false;
 private boolean isImported = false;

 public Book(String description, int qty, double price) {
 super(description, qty, price);
 }

 public boolean isTaxed() {
 return isTaxed;
 }

 public boolean isImported() {
 return isImported;
 }

 public void setImported(boolean b) {
 isImported = b;
 }
}
```

```

public class CD extends Goods {
 private boolean isTaxed = true;
 private boolean isImported = false;

 public CD(String description, int qty, double price) {
 super(description, qty, price);
 }

 public boolean isTaxed() {
 return isTaxed;
 }

 public boolean isImported() {
 return isImported;
 }

 public void setImported(boolean b) {
 isImported = b;
 }
}

public class Cosmetics extends Goods {
 private boolean isTaxed = true;
 private boolean isImported = false;

 public Cosmetics(String description, int qty, double price) {
 super(description, qty, price);
 }

 public boolean isTaxed() {
 return isTaxed;
 }

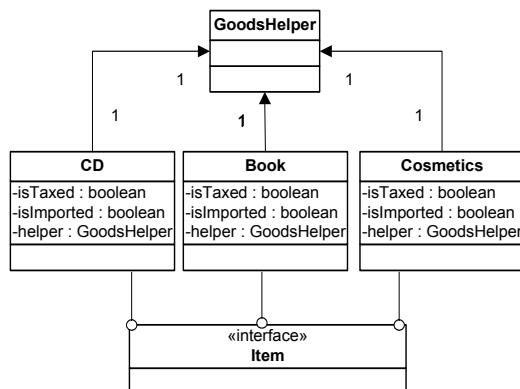
 public boolean isImported() {
 return isImported;
 }

 public void setImported(boolean b) {
 isImported = b;
 }
}

```

**Alternative solution:** Alternatively, instead of using inheritance, we can use object composition to achieve code reuse as discussed in Q8 in Java section. If you were to use object composition instead of inheritance, you would have classes *Book*, *CD* and *Cosmetics* implementing the *Item* interface directly (*Goods* class would not be required), and make use of a *GoodsHelper* class to achieve code reuse through composition.

interface inheritance where code reuse is achieved through composition [GoodsHelper]. code not shown.



Let's define a *Tax* class, which is responsible for calculating the tax. The *Tax* class is composed in your *Goods* class, which makes use of object composition to achieve code reuse.

```
public class Tax {
```

```
//stay away from hard coding values. Define constants or read from a ".properties" file
public static final double SALES_TAX = 0.10; //10%
public static final double IMPORT_TAX = 0.05; //5%

private double salesTax = 0.0;
private double importTax = 0.0;

public void calculate(boolean isTaxable, boolean isImported, double price) {
 if (isTaxable) {
 salesTax = price * SALES_TAX;
 }
 if (isImported) {
 importTax = price * IMPORT_TAX;
 }
}

public double getTotalUnitTax() {
 return this.salesTax + this.importTax;
}
```

**Factory method pattern:** To create the items shown above we could use the **factory method pattern** as described in Q46 in Java section. We would also implement the factory class as a singleton using the **singleton design pattern** as described in Q45 in Java section. The factory method design pattern **instantiates a class in a more flexible way than directly calling the constructor. It loosely couples your calling code from the items it creates like CD, Book, etc.** Let's look at why factory method pattern is more flexible.

- Sometimes factory methods have to return a single instance of a class instead of creating new objects each time or return an instance from a pool of objects.
- Factory methods have to return a subtype of the type requested. It also can request the caller to refer to the returned object by **its interface rather than by its implementation**, which enables objects to be created without making their implementation classes public.
- Sometimes old ways of creating objects can be replaced by new ways of creating the same objects or new classes can be added using polymorphism without changing any of the existing code which uses these objects. **For example:** Say you have a *Fruit* abstract class with *Mango* and *Orange* as its concrete subclasses, later on you can add an *Apple* subclass without breaking the code which uses these objects.

The factory method patterns consist of a **product class hierarchy** and a **creator class hierarchy**.

```
/**
 * ItemFactory is responsible for creating Item objects like CD, Book, and Cosmetics etc
 */
public abstract class ItemFactory {
 public abstract Item getItem(int itemType, String description, int qty, double price) throws ItemException;
}

/**
 * GoodsFactory responsible for creating Item objects like CD, Book, and Cosmetics etc
 */
public class GoodsFactory extends ItemFactory {

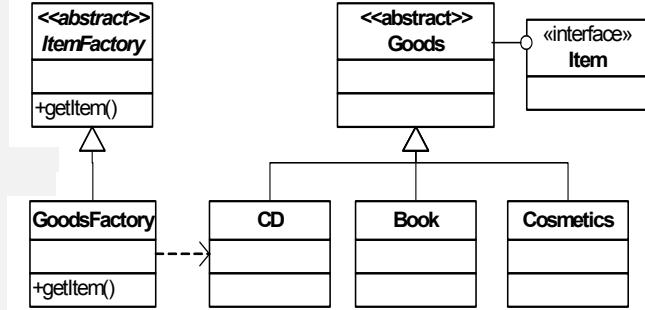
 protected GoodsFactory() {} //protected so that only ItemFactorySelector within this package can instantiate it to provide a single
 // point of access (i.e singleton).

 /**
 * Factory method, which decides how to create Items.
 *
 * Benefits are: -- loosely-couples the client (i.e ShoppingBasketBuilder class) from Items such as CD, Book, and Cosmetics etc.
 * In future if we need to create a Book item, which is imported, we can easily incorporate this by adding a new
 * item.TYPE_BOOK_IMPORTED and subsequently adding following piece of code as shown:
 *
 * else if(itemType == TYPE_BOOK_IMPORTED){
 * item = new Book(description, qty,price);
 * item.setIsImported(true);
 * }
 *
 * --It is also possible to create an object cache or object pool of our items instead of creating a new instance
```

```

* every time without making any changes to the calling class.
*
* --Java does not support overloaded constructors which take same parameter list. Instead, use several factory methods.
* E.g. getImportedItem(int itemType, String description, int qty, double price).
*/
public Item getItem(int itemType, String description, int qty, double price) throws ItemException {
 Item item = null;
 if (itemType == Item.TYPE_BOOK) {
 item = new Book(description, qty, price);
 } else if (itemType == Item.TYPE_CD) {
 item = new CD(description, qty, price);
 } else if (itemType == Item.TYPE_CD_IMPORTED) {
 item = new CD(description, qty, price);
 item.setImported(true);
 } else if (itemType == Item.TYPE_COSMETICS) {
 item = new Cosmetics(description, qty, price);
 } else {
 throw new ItemException("Invalid ItemType=" + itemType);
 }
 return item;
}

```



Let's use the abstract factory pattern to create an *ItemFactory* and the singleton pattern to provide a single point of access to the *ItemFactory* returned.

**Abstract factory pattern:** This pattern is one level of abstraction higher than the factory method pattern because you have an abstract factory (or factory interface) and have multiple concrete factories. Abstract factory pattern usually has a specific method for each concrete type being returned (e.g. `createCircle()`, `createSquare()` etc). Alternatively you can have a single method e.g. `createShape(...)`.

**Singleton pattern:** Ensures that a class has only one instance and provides a global point of access to it (Refer Q45 in Java section). E.g. a *DataSource* should have only a single instance where it will supply multiple connections from its single *DataSource* pool.

```

/*
 * Abstract factory class which creates a singleton ItemFactory dynamically based on factory name supplied.
 * Benefits of singleton: -- single instance of the ItemFactory -- single point of access (global access within the JVM and the class loader)
 */
public class ItemFactorySelector {
 private static ItemFactory objectFactorySingleInstance = null;
 private static final String FACTORY_NAME = "com.item.GoodsFactory";

 public static ItemFactory getItemFactory() {
 try {
 if (objectFactorySingleInstance == null) {
 //Dynamically instantiate factory and factory name can also be read from a properties file.
 //In future if we need a CachedGoodsFactory which caches Items to improve memory usage then
 //we can modify the FACTORY_NAME to "com.item.CachedGoodsFactory" or conditionally select one of many factories.
 Class klassFactory = Class.forName(FACTORY_NAME);
 objectFactorySingleInstance = (ItemFactory) klassFactory.newInstance();
 }
 } catch (ClassNotFoundException cnf) {
 throw new RuntimeException("Cannot create the ItemFactory: " + cnf.getMessage());
 } catch (IllegalAccessException iae) {
 throw new RuntimeException("Cannot create the ItemFactory: " + iae.getMessage());
 } catch (InstantiationException ie) {
 throw new RuntimeException("Cannot create the ItemFactory: " + ie.getMessage());
 }
 return objectFactorySingleInstance;
 }
}

```

Now we should build a more complex shopping basket object step-by-step, which is responsible for building a basket with items like *CD*, *Book* etc and calculating total tax for the items in the basket. The **builder design pattern** is used to define the interface *ItemBuilder* and the concrete class, which implements this interface, is named *ShoppingBasketBuilder*.

**Builder pattern:** The subtle difference between the builder pattern and the factory pattern is that in builder pattern, the user is given the **choice to create the type of object he/she wants but the construction process is the same**. But

with the factory method pattern the **factory decides how to create one of several possible classes based on data provided to it.**

```
...//package & import statements

public interface ItemBuilder {
 public void buildBasket(int itemType, String description, int qty, double unit_price) throws ItemException;
 public double calculateTotalTax() throws ItemException;
 public double calculateTotal() throws ItemException;
 public void printExtendedTaxedPrice() throws ItemException;
 public Iterator getItemIterator();
}

...//package & import statements

<**
 * Builder pattern: To simplify complex object creation by defining a class whose purpose is to build instances of another class.
 * There is a subtle difference between a builder pattern and the factory pattern is that in builder pattern, the user is given
 * the choice to create the type of object he/she wants but the construction process is the same. But with the factory method
 * pattern the factory decides how to create one of several possible classes based on data provided to it.
 */
public class ShoppingBasketBuilder implements ItemBuilder {

 private List listItems = null;

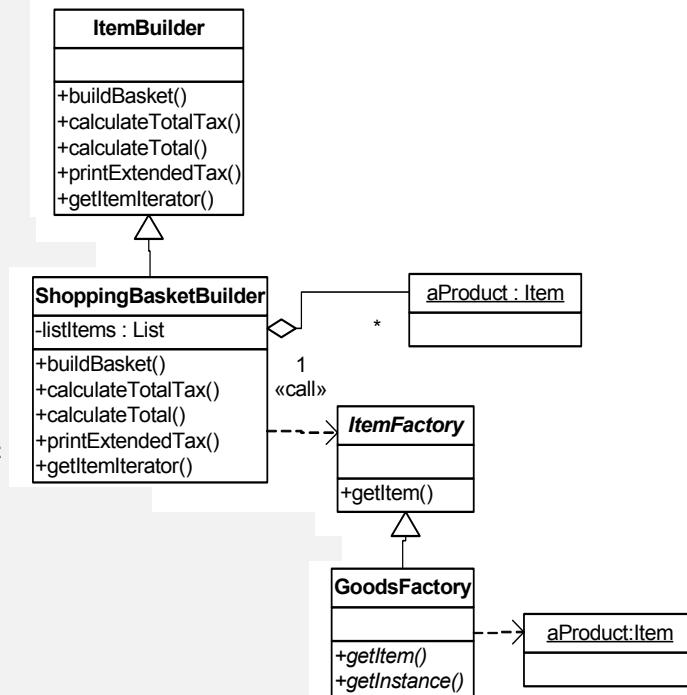
 private void addNewItem(Item item) {
 if (listItems == null) {
 listItems = new ArrayList(20);
 }
 listItems.add(item);
 }

 /**
 * builds a shopping basket
 */
 public void buildBasket(int itemType, String description, int qty, double unit_price) throws ItemException {
 //get the single instance of GoodsFactory using the singleton pattern
 //no matter how many times you call getInstance() you get access to the same instance.
 ItemFactory factory = ItemFactorySelector.getItemFactory();

 //use factory method pattern to create item objects, based on itemType supplied to it.
 Item item = factory.getItem(itemType, description, qty, unit_price);
 this.addNewItem(item);
 }

 /**
 * calculates total tax on the items in the built basket
 */
 public double calculateTotalTax() throws ItemException {
 if (listItems == null) {
 throw new ItemException("No items in the basket");
 }
 double totalTax = 0.0;
 Iterator it = listItems.iterator();
 while (it.hasNext()) {
 Item item = (Item) it.next();
 totalTax += item.getExtendedTax();
 }
 return totalTax;
 }

 /**
 * calculates total price on the items in the built basket
 */
 public double calculateTotal() throws ItemException {
 if (listItems == null) {
 throw new ItemException("No items in the basket");
 }
 double total = 0.0;
 Iterator it = listItems.iterator();
 while (it.hasNext()) {
 Item item = (Item) it.next();
 total += item.getExtendedTaxPrice();
 }
 }
}
```



```

 }
 return total;
}

/**
 * prints individual prices of the items in the built basket
 */
public void printExtendedTaxedPrice() throws ItemException {
 if (listItems == null) {
 throw new ItemException("No items in the basket");
 }
 double totalTax = 0.0;
 Iterator it = listItems.iterator();
 while (it.hasNext()) {
 Item item = (Item) it.next();
 System.out.println(item + " " + item.getExtendedTaxPrice());
 }
}

public Iterator getIterator() {
 return listItems.iterator();
}
}

```

Finally, the calling-code, which makes use of our shopping basket builder pattern to build the shopping basket step-by-step and also calculates the taxes and the grand total for the items in the shopping basket.

```

...//package & import statements

public class Shopping {
 /**
 * Class with main(String[]) method which initially gets loaded by the
 * class loader. Subsequent classes are loaded as they are referenced in the program.
 */
 public static void main(String[] args) throws ItemException {
 process();
 }

 public static void process() throws ItemException {
 //-----creational patterns: singleton, factory method and builder design patterns-----
 System.out.println("---create a shopping basket with items ---");
 //Shopping basket using the builder pattern
 ItemBuilder builder = new ShoppingBasketBuilder();
 //build basket of items using a builder pattern
 builder.buildBasket(Item.TYPE_BOOK, "Book - IT", 1, 12.00);
 builder.buildBasket(Item.TYPE_CD, "CD - JAZZ", 1, 15.00);
 builder.buildBasket(Item.TYPE_COSMETICS, "Cosmetics - Lipstick", 1, 1.0);

 //lets print prices and taxes of this built basket
 double totalTax = builder.calculateTotalTax();
 builder.printExtendedTaxedPrice();
 System.out.println("Sales Taxes: " + totalTax);
 System.out.println("Grand Total: " + builder.calculateTotal());
 System.out.println("---- After adding an imported CD to the basket ----");

 //Say now customer decides to buy an additional imported CD
 builder.buildBasket(Item.TYPE_CD_IMPORTED, "CD - JAZZ IMPORTED", 1, 15.00);

 //lets print prices and taxes of this built basket with imported CD added
 totalTax = builder.calculateTotalTax();
 builder.printExtendedTaxedPrice();
 System.out.println("Sales Taxes: " + totalTax);
 System.out.println("Grand Total: " + builder.calculateTotal());
 }
}

```

Running the above code produces an output of:

```

---create a shopping basket with items ---
1 Book - IT : 12.0
1 CD - JAZZ : 16.5
1 Cosmetics - Lipstick : 1.1
Sales Taxes: 1.6

```

Grand Total: 29.6

----- After adding an imported CD to the basket -----

```
1 Book - IT : 12.0
1 CD - JAZZ : 16.5
1 Cosmetics - Lipstick : 1.1
1 CD - JAZZ IMPORTED : 17.25
Sales Taxes: 3.85
Grand Total: 46.85
```

**Scenario:** The XYZ Retail wants to evaluate a strategy to determine items with description longer than 15 characters because it won't fit in the invoice and items with description starting with "CD" to add piracy warning label.

**Solution:** You can implement evaluating a strategy to determine items with description longer than 15 characters and description starting with "CD" applying the **strategy design pattern** as shown below:

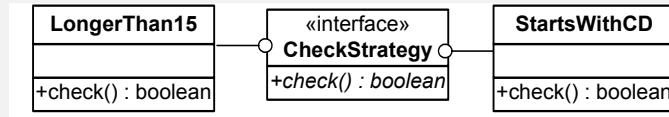
**Strategy pattern:** The Strategy pattern lets you build software as a loosely coupled collection of interchangeable parts, in contrast to a monolithic, tightly coupled system. Loose coupling makes your software much more extensible, maintainable, and reusable. The main attribute of this pattern is that each strategy encapsulates algorithms i.e. it is **not data based but algorithm based**. Refer **Q10, Q55** in Java section.

**Example:** You can draw borders around almost all Swing components, including panels, buttons, lists, and so on. Swing provides numerous border types for its components: bevel, etched, line, titled, and even compound. JComponent class, which acts as the base class for all Swing components by implementing functionality common to all Swing components, draws borders for Swing components, using strategy pattern.

```
public interface CheckStrategy {
 public boolean check(String word);
}
```

```
public class LongerThan15 implements CheckStrategy {
 public static final int LENGTH = 15; //constant

 public boolean check(String description) {
 if (description == null)
 return false;
 else
 return description.length() > LENGTH;
 }
}
```



```
public class StartsWithCD implements CheckStrategy {
 public static final String STARTS_WITH = "cd";

 public boolean check(String description) {
 String s = description.toLowerCase();
 if (description == null || description.length() == 0)
 return false;
 else
 return s.startsWith(STARTS_WITH);
 }
}
```

**Scenario:** The XYZ retail has decided to count the number of items, which satisfy the above strategies.

**Solution:** You can apply the **decorator design pattern** around your strategy design pattern. Refer **Q20** in Java section for the decorator design pattern used in `java.io.*`. The decorator acts as a **wrapper** around the `CheckStrategy` objects where by call the `check(...)` method on the `CheckStrategy` object and if it returns true then increment the counter. The decorator design pattern can be used to provide additional functionality to an object of some kind. The key to a decorator is that a decorator "wraps" the object decorated and looks to a client exactly the same as the object wrapped. This means that the **decorator implements the same interface as the object it decorates**.

**Decorator design pattern:** You can think of a decorator as a shell around the object decorated. The decorator catches any message that a client sends to the object instead. The decorator may apply some action and then pass the message it received on to the decorated object. That object probably returns a value to the decorator which may again apply an action to that result, finally sending the (perhaps-modified) result to the original client. To the client the **decorator is invisible**. It just sent a message and got a result. However the decorator had two chances to enhance the result returned.

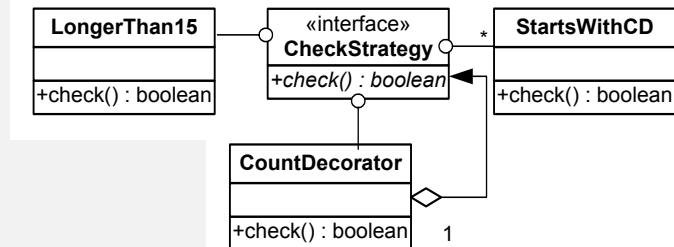
```
public class CountDecorator implements CheckStrategy {
 private CheckStrategy cs = null;
 private int count = 0;

 public CountDecorator(CheckStrategy cs) {
 this.cs = cs;
 }

 public boolean check(String description) {
 boolean isFound = cs.check(description);
 if (isFound)
 this.count++;
 return isFound;
 }

 public int count() {
 return this.count;
 }

 public void reset() {
 this.count = 0;
 }
}
```



There is a subtle difference between the decorator pattern and the proxy pattern is that, the main intent of the decorator pattern is to enhance the functionality of the target object whereas the main intent of the proxy pattern is to control access to the target object.

A decorator object's interface must conform to the interface of the component it decorates

Now, let's see the calling class *Shopping*:

```
//.... package & import statements

public class Shopping {
//...

 public static void process() throws ItemException {
 ...
 Iterator it = builder.getIterator();
 boolean bol = false;
 CheckStrategy strategy = null;

 it = builder.getIterator();
 //for starting with CD
 strategy = new StartsWithCD();
 strategy = new CountDecorator(strategy);
 while (it.hasNext()) {
 Item item = (Item) it.next();
 bol = strategy.check(item.getDescription());
 System.out.println("\n" + item.getDescription() + " --> " + bol);
 }

 System.out.println("No of descriptions starts with CD -->" + ((CountDecorator) strategy).count());

 it = builder.getIterator();
 //for starting with CD
 strategy = new LongerThan15();
 strategy = new CountDecorator(strategy);
 while (it.hasNext()) {
 Item item = (Item) it.next();
 bol = strategy.check(item.getDescription());
 System.out.println("\n" + item.getDescription() + " --> " + bol);
 }

 System.out.println("No of descriptions longer than 15 characters -->" + ((CountDecorator) strategy).count());
 }
}
```

Running the above code produces an output of:

```
---count item description starting with 'cd' or longer than 15 characters ---
----- description satarting with cd -----
Book - IT --> false
CD - JAZZ --> true
Cosmetics - Lipstick --> false
CD - JAZZ IMPORTED --> true
No of descriptions starts with CD -->2
----- description longer than 15 characters -----
```

```
Book - IT --> false
CD - JAZZ --> false
Cosmetics - Lipstick --> true
CD - JAZZ IMPORTED --> true
No of descriptions longer than 15 characters -->2
```

**Scenario:** So far so good, for illustration purpose if you need to adapt the strategy class to the *CountDecorator* class so that you do not have to explicitly cast your strategy classes to *CountDecorator* as shown in bold arrow in the class *Shopping*. We can overcome this by slightly rearranging the classes. The class *CountDecorator* has two additional methods *count()* and *reset()*. If you only just add these methods to the interface *CheckStrategy* then the classes *LongerThan15* and *StartsWithCD* should provide an implementation for these two methods. These two methods make no sense in these two classes.

**Solution:** So, to overcome this you can introduce an adapter class named *CheckStrategyAdapter*, which just provides a bare minimum default implementation.

```
public interface CheckStrategy {
 public boolean check(String word);
 public int count();
 public void reset();
}

/**
 * This is an adapter class which provides default implementations to be extended not to be used and facilitates its subclasses to be
 * adapted to each other. Throws an unchecked exception to indicate improper use.
 */

public class CheckStrategyAdapter implements CheckStrategy {
 public boolean check(String word) {
 throw new RuntimeException("Improper use of CheckStrategyAdapter class method check(String word)");
 }

 public int count() {
 throw new RuntimeException("Improper use of CheckStrategyAdapter class method count()");
 }

 public void reset() {
 throw new RuntimeException("Improper use of CheckStrategyAdapter class method reset()");
 }
}

public class LongerThan15 extends CheckStrategyAdapter {
 public static final int LENGTH = 15;

 public boolean check(String description) {
 if (description == null)
 return false;
 else
 return description.length() > LENGTH;
 }
}

public class StartsWithCD extends CheckStrategyAdapter {
 public static final String STARTS_WITH = "cd";

 public boolean check(String description) {
 String s = description.toLowerCase();
 if (description == null || description.length() == 0)
 return false;
 else
 return s.startsWith(STARTS_WITH);
 }
}

public class CountDecorator extends CheckStrategyAdapter {

 private CheckStrategy cs = null;
 private int count = 0;

 public CountDecorator(CheckStrategy cs) {
 this.cs = cs;
 }
}
```

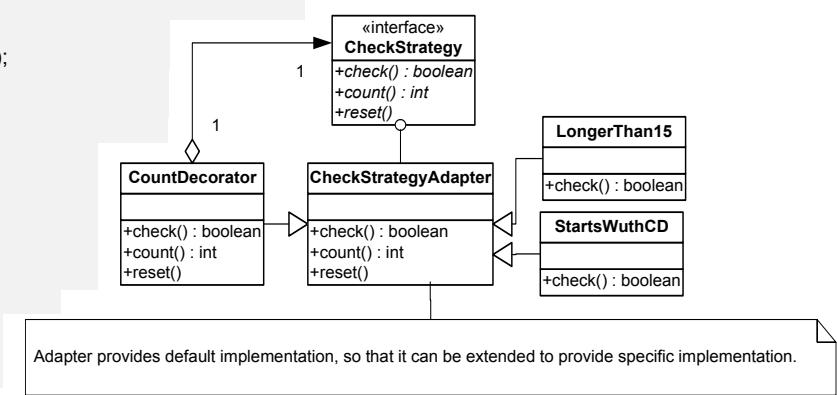
```

public boolean check(String description) {
 boolean isFound = cs.check(description);
 if (isFound)
 this.count++;
 return isFound;
}

public int count() {
 return this.count;
}

public void reset() {
 this.count = 0;
}
}

```



Now, let's see the revised calling class **Shopping**:

```

//...package & import statements

public class Shopping {
//.....
public static void process() throws ItemException {

-----Strategy and decorator design pattern-----
System.out.println("----count item description starting with 'cd'or longer than 15 characters ---");
Iterator it = builder.getIterator();
boolean bol = false;
CheckStrategy strategy = null;
System.out.println("----- description satarting with cd -----");
it = builder.getIterator();
//for starting with CD
strategy = new StartsWithCD();
strategy = new CountDecorator(strategy);
while (it.hasNext()) {
 Item item = (Item) it.next();
 bol = strategy.check(item.getDescription());
 System.out.println(item.getDescription() + " --> " + bol);
}

System.out.println("No of descriptions starts with CD -->" + strategy.count());

System.out.println("----- description longer than 15 characters -----");
it = builder.getIterator();
//for starting with CD
strategy = new LongerThan15();
strategy = new CountDecorator(strategy);
while (it.hasNext()) {
 Item item = (Item) it.next();
 bol = strategy.check(item.getDescription());
 System.out.println(item.getDescription() + " --> " + bol);
}
System.out.println("No of descriptions longer than 15 characters -->" + strategy.count());
}
}

```

The output is:

```

----count item description starting with 'cd'or longer than 15 characters ---
----- description satarting with cd -----
Book - IT --> false
CD - JAZZ --> true
Cosmetics - Lipstick --> false
CD - JAZZ IMPORTED --> true
No of descriptions starts with CD -->2

----- description longer than 15 characters -----

Book - IT --> false
CD - JAZZ --> false
Cosmetics - Lipstick --> true
CD - JAZZ IMPORTED --> true

```

No of descriptions longer than 15 characters ->2

**Scenario:** The XYZ Retail also requires a piece of code, which performs different operations depending on the type of item. If the item is an instance of *CD* then you call a method to print its catalog number. If the item is an instance of *Cosmetics* then you call a related but different method to print its colour code. If the item is an instance of *Book* then you call a separate method to print its ISBN number. One way of implementing this is using the Java constructs **instanceof** and **explicit type casting** as shown below:

```
it = builder.getIterator();
while(it.hasNext();) {
 String name = null;
 Item item = (Item)iter.next();

 if(item instanceof CD) {
 ((CD) item). markWithCatalogNumber();
 } else if (item instanceof Cosmetics) {
 ((Cosmetics) item). markWithColourCode ();
 } else if (item instanceof Book) {
 ((Book) item). markWithISBNNumber();
 }
}
```

**Solution:** The manipulation of a collection of polymorphic objects with the constructs **typecasts** and **instanceof** as shown above can get messy and unmaintainable with large elseif constructs and these constructs in frequently accessed methods/ loops can adversely affect performance. You can apply the **visitor** design pattern to avoid using these typecast and "instanceof" constructs as shown below:

**Visitor pattern:** The visitor pattern makes adding new operations easy and all the related operations are localized in a visitor. The visitor pattern allows you to manipulate a collection of polymorphic objects without the messy and unmaintainable **typecasts** and **instanceof** operations. Visitor pattern allows you to add new operations, which affect a class hierarchy without having to change any of the classes in the hierarchy. **For example** we can add a **GoodsDebugVisitor** class to have the visitor just print out some debug information about each item visited etc. In fact you can write any number of visitor classes for the *Goods* hierarchy e.g. **GoodsLabellingVisitor**, **GoodsXXXXVisitor etc.**

```
public interface Item {
 ...
 public void accept(ItemVisitor visitor);
}

public interface ItemVisitor {
 public void visit (CD cd);
 public void visit (Cosmetics cosmetics);
 public void visit (Book book);
}

/**
 * visitor class which calls different methods depending
 * on type of item.
 */
public class GoodsLabellingVisitor implements ItemVisitor {

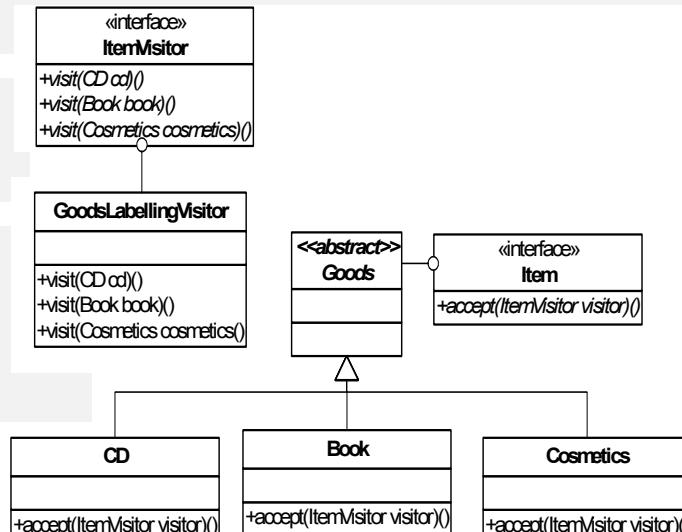
 public void visit(CD cd) {
 markWithCatalogNumber(cd);
 }

 public void visit(Cosmetics cosmetics) {
 markWithColorNumber(cosmetics);
 }

 public void visit(Book book) {
 markWithISBNNumber(book);
 }

 private void markWithCatalogNumber(CD cd) {
 System.out.println("Catalog number for : " + cd.getDescription());
 }

 private void markWithColorNumber(Cosmetics cosmetics) {
```



```

 System.out.println("Color number for : " + cosmetics.getDescription());
 }

 public void markWithISBNNumber(Book book) {
 System.out.println("ISBN number for : " + book.getDescription());
 }
}

public class CD extends Goods {
//...
 public void accept(ItemVisitor visitor) {
 visitor.visit(this);
 }
}

public class Book extends Goods {
//...
 public void accept(ItemVisitor visitor) {
 visitor.visit(this);
 }
}

public class Cosmetics extends Goods {
//...
 public void accept(ItemVisitor visitor) {
 visitor.visit(this);
 }
}

```

Now, let's see the calling code or class *Shopping*:

```

//... package and import statements

public class Shopping {

 public static void process() throws ItemException {

 //visitor pattern example, no messy instanceof and typecast constructs
 it = builder.getIterator();
 ItemVisitor visitor = new GoodsLabellingVisitor ();
 while (it.hasNext()) {
 Item item = (Item) it.next();
 item.accept(visitor);
 }
 }
}

```

The output is:

```

---- markXXXX(): avoid huge if else statements, instanceof & type casts -----
ISBN number for : Book - IT
Catalog number for : CD - JAZZ
Color number for : Cosmetics - Lipstick
Catalog number for : CD - JAZZ IMPORTED

```

**Scenario:** The XYZ Retail would like to have a functionality to iterate through every second or third item in the basket to randomly collect some statistics on price.

**Solution:** This can be implemented by applying the **iterator design pattern**.

**Iterator pattern:** Provides a way to access the elements of an aggregate object without exposing its underlying implementation.

```

//... package and import statements

public interface ItemBuilder {
//..
 public com.item.Iterator getItemIterator();
}

package com.item;

```

```
public interface Iterator {
 public Item nextItem();
 public Item previousItem();
 public Item currentItem();
 public Item firstItem();
 public Item lastItem();
 public boolean isDone();
 public void setStep(int step);
}
```

//... package and import statements

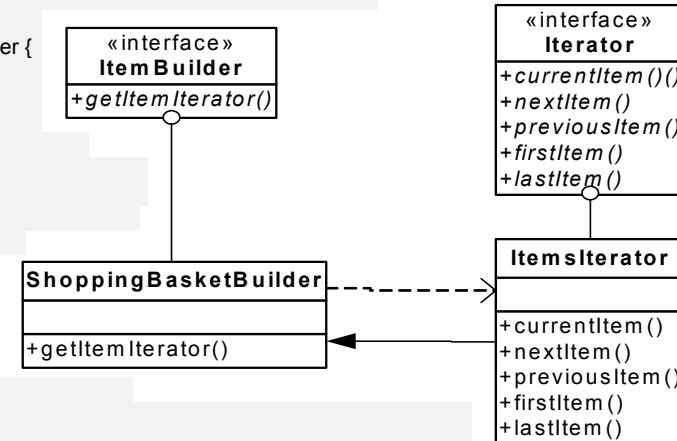
```
public class ShoppingBasketBuilder implements ItemBuilder {
 private List listItems = null;

 public Iterator getIterator() {
 return listItems.iterator();
 }

 public com.item.Iterator getItemIterator() {
 return new ItemsIterator();
 }

 /**
 * inner class which iterates over basket of items
 */
 class ItemsIterator implements com.item.Iterator {
 private int current = 0;

 private int step = 1;
```



```

 public Item nextItem() {
 Item item = null;
 current += step;
 if (!isDone()) {
 item = (Item) listItems.get(current);
 }
 return item;
 }

 public Item previousItem() {
 Item item = null;
 current -= step;
 if (!isDone()) {
 item = (Item) listItems.get(current);
 }
 return item;
 }

 public Item firstItem() {
 current = 0;
 return (Item) listItems.get(current);
 }

 public Item lastItem() {
 current = listItems.size() - 1;
 return (Item) listItems.get(current);
 }

 public boolean isDone() {
 return current >= listItems.size() ? true : false;
 }
```

```

 public Item currentItem() {
 if (!isDone()) {
 return (Item) listItems.get(current);
 } else {
 return null;
 }
 }
```

```

public void setStep(int step) {
 this.step = step;
}
}
}

```

Now, let's see the calling code *Shopping*:

```

//... package & import statements

public class Shopping {
 //..

 public static void process() throws ItemException {
 //Iterator pattern example, inner implementations of ShopingBasketBuilder is protected.
 com.item.Iterator itemIterator = builder.getItemIterator();

 //say we want to traverse through every second item in the basket
 itemIterator.setStep(2);
 Item item = null;
 for (item = itemIterator.firstItem(); !itemIterator.isDone(); item = itemIterator.nextItem()) {
 System.out.println("nextItem: " + item.getDescription() + " =====> " + item.getExtendedTaxPrice());
 }

 item = itemIterator.lastItem();
 System.out.println("lastItem: " + item.getDescription() + " =====> " + item.getExtendedTaxPrice());

 item = itemIterator.previousItem();
 System.out.println("previousItem : " + item.getDescription() + "=====>" + item.getExtendedTaxPrice());
 }
}

```

The output is:

```

----- steps through every 2nd item in the basket -----
nextItem: Book - IT =====> 12.0
nextItem: Cosmetics - Lipstick =====> 1.1
lastItem: CD - JAZZ IMPORTED =====> 17.25
previousItem : CD - JAZZ=====>16.5

```

**Scenario:** The XYZ Retail buys the items in bulk from warehouses and sells them in their retail stores. All the items sold need to be prepared for retail prior to stacking in the shelves for trade. The preparation involves 3 steps for all types of items, i.e. adding the items to stock in the database, applying barcode to each item and finally marking retail price on the item. The preparation process is common involving 3 steps but each of these individual steps is specific to type of item i.e. *Book*, *CD*, and *Cosmetics*.

**Solution:** The above functionality can be implemented applying the template method design pattern as shown below:

**Template method pattern:** When you have a sequence of steps to be processed within a method and you want to defer some of the steps to its subclass then you can use a template method pattern. So the template method lets the subclass to redefine some of the steps.

**Example** Good example of this is the *process()* method in the Struts *RequestProcessor* class, which executes a sequence of *processXXXX(...)* methods allowing the subclass to override some of the methods when required. Refer **Q110** in Enterprise section.

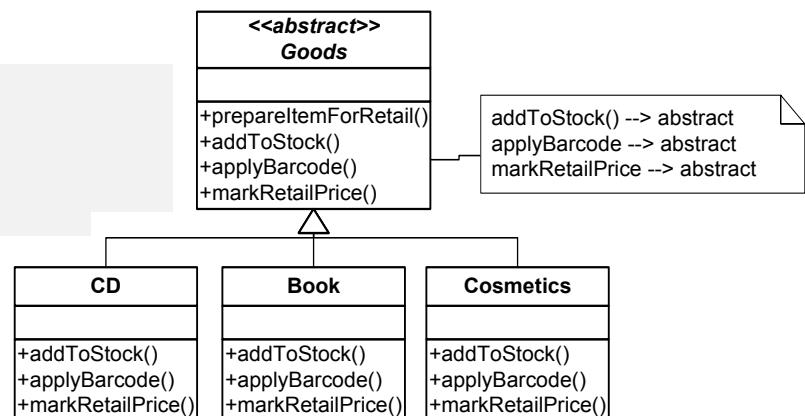
```

//...
public abstract class Goods implements Item {
 //...

 /**
 * The template method
 */
 public void prepareItemForRetail() {
 addToStock();
 applyBarcode();
 markRetailPrice();
 }

 public abstract void addToStock();
}

```



```

public abstract void applyBarcode();
public abstract void markRetailPrice();

}

//..
public class Book extends Goods {
//..

//following methods gets called by the template method

public void addToStock() {
 //database call logic to store the book in stock table.
 System.out.println("Book added to stock : " + this.getDescription());
}

public void applyBarcode() {
 //logic to print and apply the barcode to book.
 System.out.println("Bar code applied to book : " + this.getDescription());
}

public void markRetailPrice() {
 //logic to read retail price from the book table and apply the retail price.
 System.out.println("Mark retail price for the book : " + this.getDescription());
}

}

//..

public class CD extends Goods {
//..

//following methods gets called by the template method

public void addToStock() {
 //database call logic to store the cd in stock table.
 System.out.println("CD added to stock : " + this.getDescription());
}

public void applyBarcode() {
 //logic to print and apply the barcode to cd.
 System.out.println("Bar code applied to cd : " + this.getDescription());
}

public void markRetailPrice() {
 //logic to read retail price from the cd table and apply the retail price.
 System.out.println("Mark retail price for the cd : " + this.getDescription());
}

}

//..

public class Cosmetics extends Goods {
//..

public void addToStock() {
 //database call logic to store the cosmetic in stock table.
 System.out.println("Cosmetic added to stock : " + this.getDescription());
}

public void applyBarcode() {
 //logic to print and apply the barcode to cosmetic.
 System.out.println("Bar code applied to cosmetic : " + this.getDescription());
}

public void markRetailPrice() {
 //logic to read retail price from the cosmetic table and apply the retail price.
 System.out.println("Mark retail price for the cosmetic : " + this.getDescription());
}

}

```

Now, let's see the calling code *Shopping*:

```

//...
public class Shopping {
//...

```

```
public static void process() throws ItemException {
 //...

 Item item = null;
 for (item = itemIterator.firstItem(); !itemIterator.isDone(); item = itemIterator.nextItem()) {
 item.prepareItemForRetail();
 System.out.println("-----");
 }
}
```

The output is:

----- prepareItemForRetail() -----  
Book added to stock : Book - IT  
Bar code applied to book : Book - IT  
Mark retail price for the book : Book - IT

**Scenario:** The employees of XYZ Retail are at various positions. In a hierarchy, the general manager has subordinates, and also the sales manager has subordinates. The retail sales staffs have no subordinates and they report to their immediate manager. The company needs functionality to calculate salary at different levels of the hierarchy.

**Solution:** You can apply the **composite design pattern** to represent the XYZ Retail company employee hierarchy.

**Composite design pattern:** The composite design pattern composes objects into tree structures where individual objects like sales staff and composite objects like managers are handled uniformly. Refer [Q52](#) in Java section or [Q25](#) in Enterprise section.

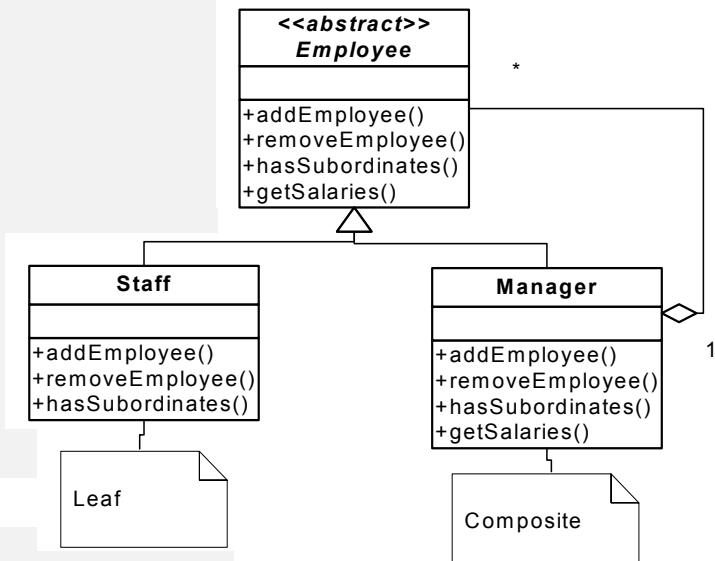
```
/**
 * Base employee class
 */
public abstract class Employee {
 private String name;
 private double salary;

 public Employee(String name, double salary) {
 this.name = name;
 this.salary = salary;
 }

 public String getName() {
 return name;
 }

 public double getSalaries() {
 return salary;
 }

 public abstract boolean addEmployee(Employee emp);
 public abstract boolean removeEmployee(Employee emp);
 protected abstract boolean hasSubordinates();
}
```



```

 return subordinates.add(emp);
 }

public boolean removeEmployee(Employee emp) {
 if (subordinates == null) {
 subordinates = new ArrayList(10);
 }
 return subordinates.remove(emp);
}

/**
 * Recursive method call to calculate the sum of salary of a manager and his subordinates, which means sum of salary of a manager
 * on whom this method was invoked and any employees who themselves will have any subordinates and so on.
 */
public double getSalaries() {
 double sum = super.getSalaries(); //this one's salary

 if (this.hasSubordinates()) {
 for (int i = 0; i < subordinates.size(); i++) {
 sum += ((Employee) subordinates.get(i)).getSalaries();
 }
 }
 return sum;
}

public boolean hasSubordinates() {
 boolean hasSubordinates = false;
 if (subordinates != null && subordinates.size() > 0) {
 hasSubordinates = true;
 }
 return hasSubordinates;
}

/**
 * This is the leaf staff employee object. staff do not have any subordinates.
 */
public class Staff extends Employee {

 public Staff(String name, double salary) {
 super(name, salary);
 }

 public boolean addEmployee(Employee emp) {
 throw new RuntimeException("Improper use of Staff class");
 }

 public boolean removeEmployee(Employee emp) {
 throw new RuntimeException("Improper use of Staff class");
 }

 protected boolean hasSubordinates() {
 return false;
 }
}

```

Now, let's see the calling code *Shopping*:

```

//...
public class Shopping {
//.....

public static void process() throws ItemException {
//.....

System.out.println("----- Employee hierachy & getSalaries() recursively -----");
//Employee hierachy

Employee generalManager = new Manager("John Smith", 100000.00);

Employee salesManger = new Manager("Peter Rodgers", 80000.00);
Employee logisticsManger = new Manager("Graham anthony", 90000.00);

```

```

Employee staffSales1 = new Staff("Lisa john", 40000.00);
Employee staffSales2 = new Staff("Pamela watson", 50000.00);
salesManger.addEmployee(staffSales1);
salesManger.addEmployee(staffSales2);

Employee logisticsTeamLead = new Manager("Cooma kumar", 70000.00);

Employee staffLogistics1 = new Staff("Ben Sampson", 60000.00);
Employee staffLogistics2 = new Staff("Vincent Chou", 20000.00);
logisticsTeamLead.addEmployee(staffLogistics1);
logisticsTeamLead.addEmployee(staffLogistics2);

logisticsManger.addEmployee(logisticsTeamLead);

generalManager.addEmployee(salesManger);
generalManager.addEmployee(logisticsManger);

System.out.println(staffSales1.getName() + "-->" + staffSales1.getSalaries());
System.out.println(staffSales2.getName() + "-->" + staffSales2.getSalaries());

System.out.println("Logistics dept " + "-->" + logisticsManger.getSalaries());

System.out.println("General Manager " + "-->" + generalManager.getSalaries());
}
}

```

The output is:

```

----- Employee hierarchy & getSalaries() recursively -----
Lisa john-->40000.0
Pamela watson-->50000.0
Logistics dept --> 240000.0
General Manager --> 510000.0

```

**Scenario:** The purchasing staffs (aka logistics staff) of the XYZ Retail Company need to **interact with other subsystems** in order to place purchase orders. They need to communicate with their stock control department to determine the stock levels, also need to communicate with their wholesale supplier to determine availability of stock and finally with their bank to determine availability of sufficient funds to make a purchase.

**Solution:** You can apply the façade design pattern to implement the above scenario.

**Façade pattern:** The façade pattern provides an interface to large subsystems of classes. A common design goal is to minimize the communication and dependencies between subsystems. One way to achieve this goal is to introduce a façade object that provides a single, simplified interface.

```

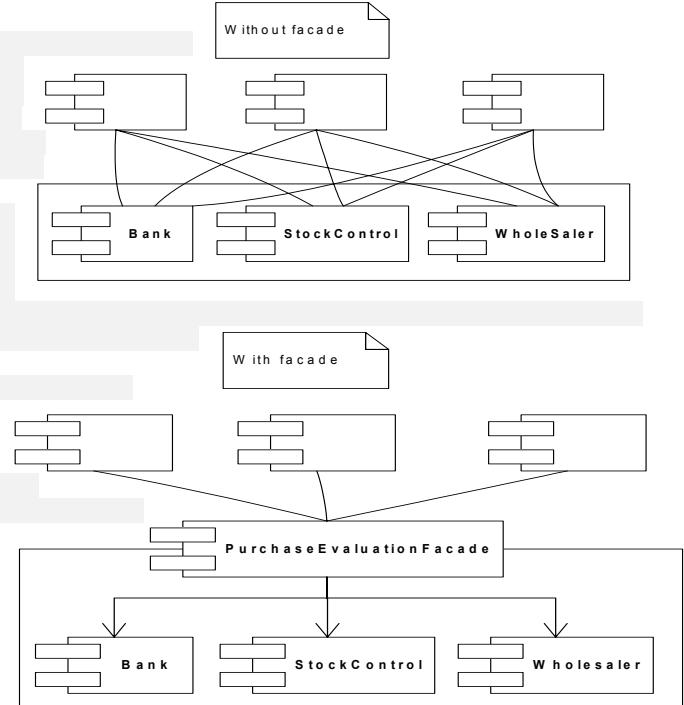
public class StockControl {
 public boolean isBelowReorderpoint(Item item) {
 //logic to evaluate stock level for item
 return true;
 }
}

public class Bank {
 public boolean hasSufficientFunds() {
 //logic to evaluate if we have sufficient savings goes here
 return true;
 }
}

public class WholeSaler {
 public boolean hasSufficientStock(Item item) {
 //logic to evaluate if the wholesaler has enough stock goes here
 return true;
 }
}

/**
 * This is the facade class
 */
public class PurchaseEvaluation {
 private StockControl stockControl = new StockControl();
}

```



```

private WholeSaler wholeSaler = new WholeSaler();
private Bank bank = new Bank();

public boolean shouldWePlaceOrder(Item item) {
 if (!stockControl.isBelowReorderpoint(item)) {
 return false;
 }

 if (!wholeSaler.hasSufficientStock(item)) {
 return false;
 }

 if (!bank.hasSufficientFunds()) {
 return false;
 }

 return true;
}

```

Now, let's see the calling code or class *Shopping*:

```

//.....
public class Shopping {
//.....
public static void process() throws ItemException {
//....
//-----facade design pattern -----
System.out.println("-----shouldWePlaceOrder-----");
PurchaseEvaluation purchaseEval = new PurchaseEvaluation();
boolean shouldWePlaceOrder = purchaseEval.shouldWePlaceOrder(item);
System.out.println("shouldWePlaceOrder=" + shouldWePlaceOrder);
}
}

```

The output is:

```

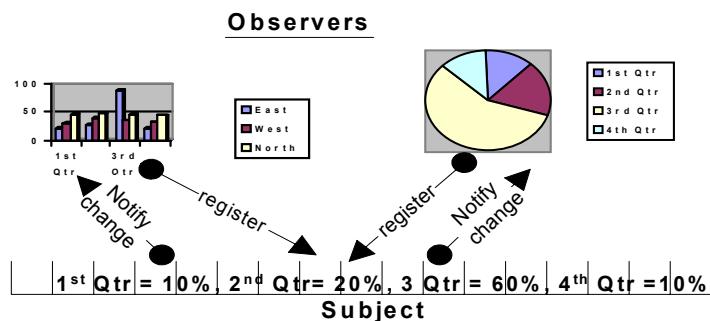
-----shouldWePlaceOrder-----
shouldWePlaceOrder=true

```

**Scenario:** The purchasing department also requires functionality where, when the stock control system is updated, all the registered departmental systems like logistics and sales should be notified of the change.

**Solution:** This can be achieved by applying the observer design pattern as shown below:

**Observer pattern:** defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically. (aka publish-subscribe pattern)



```

/**
 * This is an observer (aka subscriber) interface. This gets notified through its update method.
 */
public interface Department {
 public void update(Item item, int qty);
}

```

```
public class LogisticsDepartment implements Department {
 public void update(Item item, int qty) {
 //logic to update department's stock goes here
 System.out.println("Logistics has updated its stock for " + item.getDescription() + " with qty=" + qty);
 }
}
```

```
public class SalesDepartment implements Department {
 public void update(Item item, int qty) {
 //logic to update department's stock goes here
 System.out.println("Sales has updated its stock for " + item.getDescription() + " with qty=" + qty);
 }
}
```

```
/*
 * Subject (publisher) class: when stock is updated, notifies all the
 * subscribers.
 */
public interface StockControl {
 public void notify(Item item, int qty);
 public void updateStock(Item item, int qty);
 public boolean addSubscribers(Department dept);
 public boolean removeSubscribers(Department dept);
}
```

//... package & import statements

```
/*
 * publisher (observable) class: when stock is updated
 * notifies all the subscribers.
 */
public class XYZStockControl implements StockControl{
 List listSubscribers = new ArrayList(10);

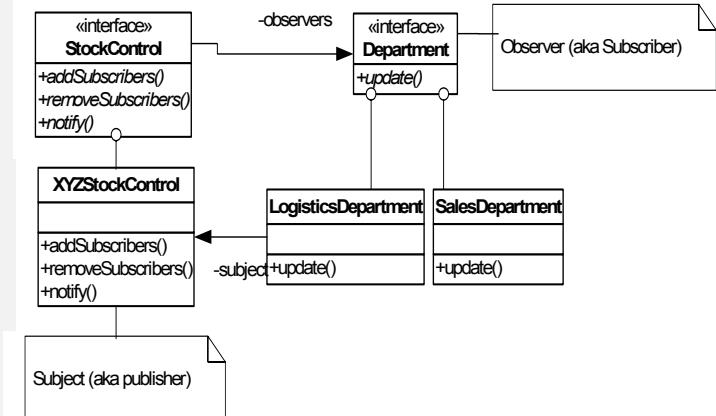
 ...

 public boolean addSubscribers(Department dept) {
 return listSubscribers.add(dept);
 }

 public boolean removeSubscribers(Department dept) {
 return listSubscribers.remove(dept);
 }

 /**
 * writes updated stock qty into databases
 */
 public void updateStock(Item item, int qty) {
 //logic to update an item's stock goes here
 notify(item, qty); //notify subscribers that with the updated stock info.
 }

 public void notify(Item item, int qty) {
 int noOfsubscribers = listSubscribers.size();
 for (int i = 0; i < noOfsubscribers; i++) {
 Department dept = (Department) listSubscribers.get(i);
 dept.update(item, qty);
 }
 }
}
```



Now, let's see the calling code or class *Shopping*:

```
// package & import statements

public class Shopping {
 //.....
 public static void process() throws ItemException {
 //.....
 //-----observer design pattern-----
 }
}
```

```
System.out.println("-----notify stock update-----");
Department deptLogistics = new LogisticsDepartment(); //observer/subscriber
Department salesLogistics = new SalesDepartment(); //observer/subscriber

StockControl stockControl = new XYZStockControl(); //observable/publisher
//let's register subscribers with the publisher
stockControl.addSubscribers(deptLogistics);
stockControl.addSubscribers(salesLogistics);

//let's update the stock value of the publisher
for (item = itemIterator.firstItem(); !itemIterator.isDone(); item = itemIterator.nextItem()) {
 if (item instanceof CD) {
 stockControl.updateStock(item, 25);
 } else if (item instanceof Book){
 stockControl.updateStock(item, 40);
 }
 else {
 stockControl.updateStock(item, 50);
 }
}
```

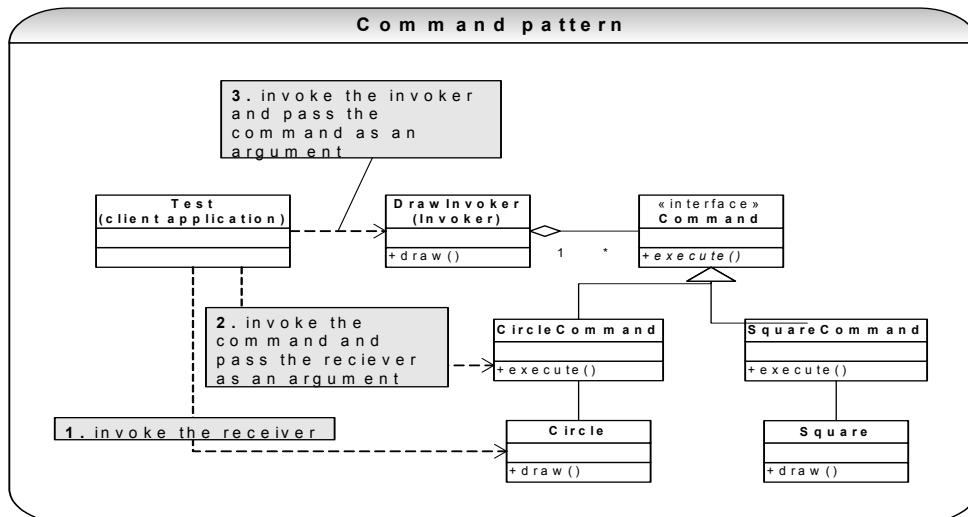
The output is:

-----notify stock update-----  
Logistics has updated its stock for Book - IT with qty=40  
Sales has updated its stock for Book - IT with qty=40  
Logistics has updated its stock for CD - JAZZ with qty=25  
Sales has updated its stock for CD - JAZZ with qty=25  
Logistics has updated its stock for Cosmetics - Lipstick with qty=50  
Sales has updated its stock for Cosmetics - Lipstick with qty=50  
Logistics has updated its stock for CD - JAZZ IMPORTED with qty=25  
Sales has updated its stock for CD - JAZZ IMPORTED with qty=25

**Scenario:** The stock control staffs require a simplified calculator, which enable them to add and subtract stock counted and also enable them to undo and redo their operations. This calculator will assist them with faster processing of stock counting operations.

**Solution:** This can be achieved by applying the **command design pattern** as shown below:

**Command pattern:** The *Command pattern* is an object behavioural pattern that allows you to achieve complete decoupling between the sender and the receiver. (A *sender* is an object that invokes an operation, and a *receiver* is an object that receives the request to execute a certain operation. With *decoupling*, the sender has no knowledge of the Receiver's interface.) The term *request* here refers to the command that is to be executed. The Command pattern also allows you to vary when and how a request is fulfilled. At times it is necessary to issue requests to objects without knowing anything about the operation being requested or the receiver of the request. In procedural languages, this type of communication is accomplished via a call-back: a function that is registered somewhere to be called at a later point. Commands are the object-oriented equivalent of call-backs and encapsulate the call-back function.



```
// package & import statements

/**
 * Invoker
 */
public class Staff extends Employee {

 private Calculator calc = new Calculator();
 private List listCommands = new ArrayList(15);
 private int current = 0;

 public Staff(String name) {
 super(name);
 }

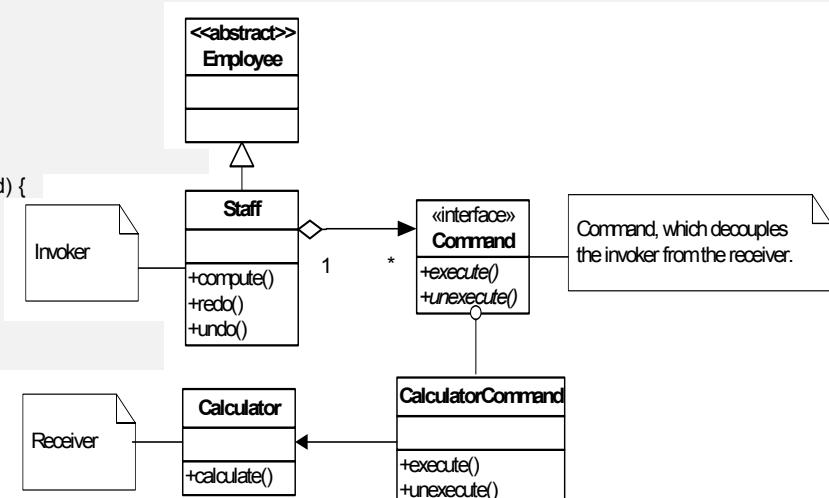
 /**
 * make use of command.
 */
 public void compute(char operator, int operand) {
 Command command = new CalculatorCommand(calc, operator, operand); //initialise the calculator
 command.execute();
 //add commands to the list so that undo operation can be performed
 listCommands.add(command);
 current++;
 }

 /**
 * perform redo operations
 */
 public void redo(int noOfLevels) {
 int noOfCommands = listCommands.size();
 for (int i = 0; i < noOfLevels; i++) {
 if (current < noOfCommands) {
 ((Command) listCommands.get(current++)).execute();
 }
 }
 }

 /**
 * perform undo operations
 */
 public void undo(int noOfLevels) {
 for (int i = 0; i < noOfLevels; i++) {
 if (current > 0) {
 ((Command) listCommands.get(--current)).unexecute();
 }
 }
 }
}
```

```
**
 * actual receiver of the command who performs calculation
 */
public class Calculator {
 private int total = 0;

 /**
 * calculates.
 */
 public void calculate(char operator, int operand) {
 switch (operator) {
 case '+':
 total += operand;
 break;
 case '-':
 total -= operand;
 break;
 }
 System.out.println("Total = " + total);
 }
}
```



```

}

/**
 * command interface
 */
public interface Command {
 public void execute();
 public void unexecute();
}

/**
 * calculator command, which decouples the receiver Calculator from the invoker staff
 */

public class CalculatorCommand implements Command {
 private Calculator calc = null;
 private char operator;
 private int operand;

 public CalculatorCommand(Calculator calc, char operator, int operand) {
 this.calc = calc;
 this.operator = operator;
 this.operand = operand;
 }

 public void execute() {
 calc.calculate(operator, operand);
 }

 public void unexecute() {
 calc.calculate(undoOperand(operator), operand);
 }

 private char undoOperand(char operator) {
 char undoOperator = ' ';
 switch (operator) {
 case '+':
 undoOperator = '-';
 break;

 case '-':
 undoOperator = '+';
 break;
 }
 return undoOperator;
 }
}

```

Now, let's see the calling code or class *Shopping*:

```

//.....
public class Shopping {
//.....
 public static void process() throws ItemException {
 //-----command design pattern-----
 System.out.println("-----Calculator with redo & undo operations-----");
 Staff stockControlStaff = new Staff("Vincent Chou");

 stockControlStaff.compute('+',10);//10
 stockControlStaff.compute('-',5);//5
 stockControlStaff.compute('+',10);//15
 stockControlStaff.compute('-',2);//13

 //lets try our undo operations
 System.out.println("-----undo operation : 1 level-----");
 stockControlStaff.undo(1);
 System.out.println("-----undo operation :2 levels-----");
 stockControlStaff.undo(2);

 //lets try our redo operations
 System.out.println("-----redo operation : 2 levels-----");
 stockControlStaff.redo(2);
 }
}

```

```

 System.out.println("-----redo operation : 1 level-----");
 stockControlStaff.redo(1);
 }
}

```

The output is:

```

-----Calculator with redo & undo operations-----
Total = 10
Total = 5
Total = 15
Total = 13
-----undo operation : 1 level-----
Total = 15
-----undo operation : 2 levels-----
Total = 5
Total = 10
-----redo operation : 2 levels-----
Total = 5
Total = 15
-----redo operation : 1 level-----
Total = 13

```

**Scenario:** The XYZ Retail has a 3<sup>rd</sup> party software component called *XYZPriceList*, which implements an interface *PriceList*. This 3<sup>rd</sup> party software component is not thread-safe. So far it performed a decent job since only the sales manager had access to this software component. The XYZ Retail now wants to provide read and write access to all the managers. The source code is not available and only the API is available, so modifying the existing component is not viable. This will cause a dirty read problem if two managers try to concurrently access this component. For example, if the sales manager tries to access an item's price while the logistics manger is modifying the price (say modification takes 1 second), then the sales manager will be reading the wrong value. Let's look at this with a code sample:

```

public interface PriceList {
 public double getPrice(int itemId);
 public void setPrice(int itemId, double newPrice);
}

//...
public class XYZPriceList implements PriceList {

 private static final Map mapPrices = new HashMap(30, .075f);
 public static PriceList singleInstance = new XYZPriceList(); //only one instance

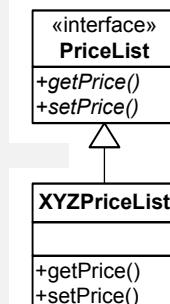
 /**
 * static initializer block
 */
 static {
 //only one item is added to keep it simple
 mapPrices.put(new Integer(1), new Double(12.00)); //Book - IT
 //... add more items to price list
 }

 public static PriceList getInstance() {
 return singleInstance;
 }

 public double getPrice(int itemId) {
 double price = ((Double)mapPrices.get(new Integer(itemId))).doubleValue();
 System.out.println("The price of the itemId " + itemId + " = " + price);
 return price;
 }

 public void setPrice(int itemId, double newPrice) {
 System.out.println("wait while mutating price from 12.0 to 15.00");
 try {
 mapPrices.put(new Integer(itemId), new Double(-1)); //transient value while updating with a proper value
 Thread.sleep(1000); //assume update/set operation takes 1 second
 mapPrices.put(new Integer(itemId), new Double(newPrice));
 } catch (InterruptedException ie) {}
 }
}

```



The multi-threaded access class:

```
public class PriceListUser implements Runnable {
 int itemId;
 double price;
 static int count = 0;

 public PriceListUser(int itemId) {
 this.itemId = itemId;
 }

 /**
 * runnable code where multi-threads are executed
 */
 public void run() {
 String name = Thread.currentThread().getName();

 if (name.equals("accessor")) {
 price = XYZPriceList.getInstance().getPrice(itemId); //using 3rd party component
 } else if (name.equals("mutator")) {
 XYZPriceList.getInstance().setPrice(itemId, 15.00); //using 3rd party component
 }
 }
}
```

Now, let's see the calling code or class *Shopping*:

```
//...
public class Shopping {
//...
public static void process() throws ItemException {
//.....
-----proxy design pattern-----
System.out.println("-----Accessing the price list-----");

PriceListUser user1 = new PriceListUser(1);//accessing same itemId=1
PriceListUser user2 = new PriceListUser(1);//accessing same itemId=1

Thread t1 = new Thread(user1);
Thread t2 = new Thread(user2);
Thread t3 = new Thread(user1);

t1.setName("accessor");//user 1 reads the price
t2.setName("mutator");//user 2 modifies the price
t3.setName("accessor");//user 1 reads the price

t1.start();//accessor user-1 reads before mutator user-2 modifies the price as 12.00
t2.start();//mutator user-2 sets the price to 15.00
t3.start();//while the user-2 is setting the price to 15.00 user-1 reads again and gets the price as 12.00
//user-2 gets the wrong price i.e gets 12.0 again instead of 15.00
}
}
```

The output is:

```
-----Accessing the price list-----
The price of the itemId 1 = 12.0
wait while mutating price from 12.0 to 15.00
The price of the itemId 1 = -1.0
OR
-----Accessing the price list-----
wait while mutating price from 12.0 to 15.00
The price of the itemId 1 = -1.0
The price of the itemId 1 = -1.0
```

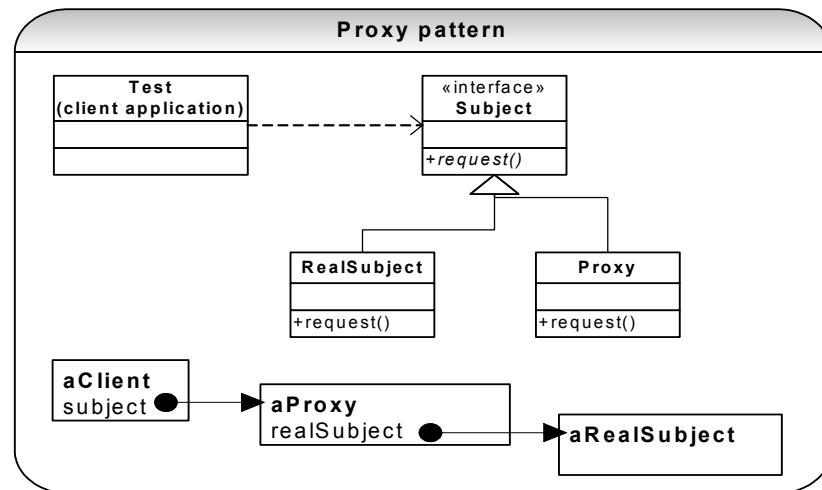
**Problem:** You get one of the two outputs shown above depending on how the threads initialized by the operating system. The first value of 12.0 is okay and the second value of 12.0 again is a **dirty read** because the value should have been modified to 15.0 by the user-2. So the user-1 reading the value for the second time should get the value of 15.0 after it has been modified.

**Solution:** This threading issue and inability to modify the existing component can be solved by applying the **proxy design pattern**. You will be writing a proxy class, which will apply the locking for the entries in the *XYZPriceList*. This proxy class internally will be making use of the *XYZPriceList* in a synchronized fashion as shown below:

**Proxy pattern:** Provides a surrogate or placeholder for another object to control access to it. Provide a surrogate or placeholder for another object to control access to it. Proxy object acts as an intermediary between the client and the target object. The proxy object has the same interface as the target object. The proxy object holds reference to the target object. There are different types of proxies:

- **Remote Proxy:** provides a reference to an object, which resides in a separate address space. e.g. EJB, RMI, CORBA etc (RMI stubs acts as a proxy for the skeleton objects.)
- **Virtual Proxy:** Allows the creation of memory intensive objects on demand. The target object will not be created until it is really needed.
- **Access Proxy:** Provides different clients with different access rights to the target object.

**Example** In Hibernate framework (Refer Q15 - Q16 in Emerging Technologies/Frameworks section) lazy loading of persistent objects are facilitated by virtual proxy pattern. Say you have a *Department* object, which has a collection of *Employee* objects. Let's say that *Employee* objects are lazy loaded. If you make a call *department.getEmployees()* then Hibernate will load only the employeeIDs and the version numbers of the *Employee* objects, thus saving loading of individual objects until later. So what you really have is a collection of proxies not the real objects. The reason being, if you have hundreds of employees for a particular department then chances are good that you will only deal with only a few of them. So, why unnecessarily instantiate all the *Employee* objects? This can be a big performance issue in some situations. So when you make a call on a particular employee i.e. *employee.getName()* then the proxy loads up the real object from the database.



```

/**
 * synchronized proxy class for XYZPriceList
 */
public class XYZPriceListProxy implements PriceList {
 //assume that we only have two items in the pricelist
 Integer[] locks = { new Integer(1), new Integer(2) }; //locks for each item in the price list

 public static PriceList singleInstance = new XYZPriceListProxy(); //single instance of XYZPriceListProxy

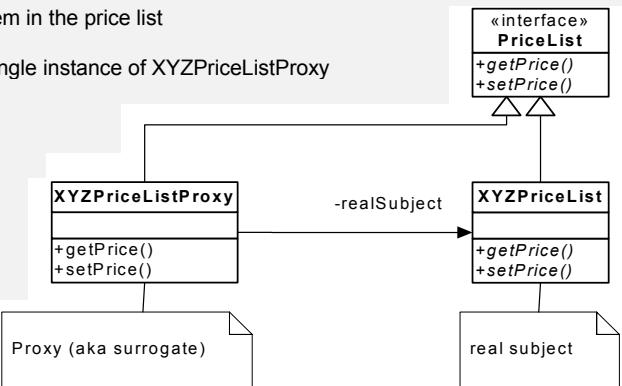
 PriceList realPriceList = XYZPriceList.getInstance(); // real object

 public static PriceList getInstance() {
 return singleInstance;
 }

 public double getPrice(int itemId) {
 synchronized (locks[itemId]) {
 return realPriceList.getPrice(itemId);
 }
 }

 public void setPrice(int itemId, double newPrice) {
 }
}

```



```
 synchronized (locks[itemId]) {
 realPriceList.setPrice(itemId, newPrice);
 }
}
```

You should make a slight modification to the *PriceListUser* class as shown below in bold.

```
public class PriceListUser implements Runnable {

 int itemId;
 double price;
 static int count = 0;

 public PriceListUser(int itemId) {
 this.itemId = itemId;
 }

 /**
 * runnable code where multi-threads are executed
 */
 public void run() {
 String name = Thread.currentThread().getName();

 if (name.equals("accessor")) {
 price = XYZPriceListProxy.getInstance().getPrice(itemId);
 } else if (name.equals("mutator")) {
 XYZPriceListProxy.getInstance().setPrice(itemId, 15.00);
 }
 }
}
```

Running the same calling code *Shopping* will render the following correct results by preventing dirty reads:

-----Accessing the price list-----  
The price of the itemId 1 = 12.0  
wait while mutating price from 12.0 to 15.00 .....  
The price of the itemId 1 = 15.0  
**OR**  
-----Accessing the price list-----  
wait while mutating price from 12.0 to 15.00 .....  
The price of the itemId 1 = 15.0  
The price of the itemId 1 = 15.0

**What is a dynamic proxy?** Dynamic proxies were introduced in J2SE 1.3, and provide an alternate dynamic mechanism for implementing many common design patterns like Façade, Bridge, Decorator, Proxy (remote proxy and virtual proxy), and Adapter. While all of these patterns can be written using ordinary classes instead of dynamic proxies, in many situations dynamic proxies are more compact and can eliminate the need for a lot of handwritten classes. Dynamic proxies are reflection-based and allow you to intercept method calls so that you can interpose additional behaviour between a class caller and its callee. Dynamic proxies are not always appropriate because this code simplification comes at a performance cost due to reflection overhead. Dynamic proxies illustrate the basics of Aspect Oriented Programming (AOP) which complements your Object Oriented Programming. Refer [Q03](#), [Q04](#) and [Q05](#) in Emerging Technologies/Frameworks section.

**Where can you use dynamic proxies?** Dynamic proxies can be used to add crosscutting concerns like logging, performance metrics, memory logging, retry semantics, test stubs, caching etc. Let's look at an example:

`InvocationHandler` interface is the heart of a proxy mechanism.

```
import java.lang.reflect.InvocationHandler;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

/**
 * Handles logging and invocation of target method
 */
public class LoggingHandler implements InvocationHandler {

 protected Object actual;

 public LoggingHandler(Object actual) {
```

```

 this.actual = actual;
 }

 public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
 try {
 System.out.println(">>>>start executing method: " + method.getName());
 Object result = method.invoke(actual, args);
 return result;
 } catch (InvocationTargetException ite) {
 throw new RuntimeException(ite.getMessage());
 } finally {
 System.out.println("<<<<finished executing method: " + method.getName());
 }
 }
}

```

Let's define the actual interface and the implementation class which adds up two integers.

```

public interface Calculator {
 public int add(int i1, int i2);
}

public class CalculatorImpl implements Calculator {

 public int add(int i1, int i2) {
 final int sum = i1 + i2;
 System.out.println("Sum is : " + sum);
 return sum;
 }
}

```

Factory method class *CalculatorFactory*, which uses the dynamic proxies when logging, is required.

```

import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Proxy;

/**
 * singleton factory
 */
public class CalculatorFactory {

 private static CalculatorFactory singleInstance = null;
 private CalculatorFactory() {}

 public static CalculatorFactory getInstance() {
 if (singleInstance == null) {
 singleInstance = new CalculatorFactory();
 }
 return singleInstance;
 }

 public Calculator getCalculator(boolean withLogging) {

 Calculator c = new CalculatorImpl();

 //use dynamic proxy if logging is required, which logs your method calls
 if (withLogging) {
 //invoke the handler, which logs and invokes the target method on the Calculator
 InvocationHandler handler = new LoggingHandler(c);

 //create a proxy
 c = (Calculator) Proxy.newProxyInstance(c.getClass().getClassLoader(), c.getClass().getInterfaces(), handler);
 }

 return c;
 }
}

```

Finally the test class:

```

import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Proxy;

```

```

public class TestProxy {

 public static void main(String[] args) {
 System.out.println("=====Without dynamic proxy=====");
 Calculator calc = CalculatorFactory.getInstance().getCalculator(false);
 calc.add(3, 2);

 System.out.println("=====With dynamic proxy=====");
 calc = CalculatorFactory.getInstance().getCalculator(true);
 calc.add(3, 2);
 }
}

```

The output is:

```

=====Without dynamic proxy=====
Sum is : 5
=====With dynamic proxy=====
>>>>start executing method: add
Sum is : 5
<<<<finished executing method: add

```

<b>Adapter pattern</b>	Sometimes a library cannot be used because its interface is not compatible with the interface required by an application. Also it is possible that we may not have the source code for the library interface. Even if we had the source code, it is not a good idea to change the source code of the library for each domain specific application. This is where you can use an adapter design pattern. Adapter lets classes work together that could not otherwise because of incompatible interfaces. This pattern is also known as a wrapper.
<b>Bridge pattern</b>	Refer Q41 in Enterprise section.
<b>Chain of responsibility pattern</b>	Refer Q22 in Enterprise section
<b>Pattern      Description</b>	
<b>J2EE Patterns</b>	
<b>MVC pattern</b>	Refer Q3 in Enterprise section Refer Q54 in Java section
<b>Front controller</b>	Refer Q24 in Enterprise section
<b>Composite View, View Helper, Dispatcher View and Service to Worker</b>	Refer Q25 in Enterprise section
<b>Business delegate</b>	Refer Q83 in Enterprise section
<b>Session façade</b>	Refer Q84 in Enterprise section
<b>Value Object</b>	Refer Q85 in Enterprise section
<b>Fast lane reader</b>	Refer Q86 in Enterprise section
<b>Service locator</b>	Refer Q87 in Enterprise section

#### Useful links:

- [http://www.allapplabs.com/Java\\_design\\_patterns/creational\\_patterns.htm](http://www.allapplabs.com/Java_design_patterns/creational_patterns.htm)
- <http://www.patterndepot.com/put/8/JavaPatterns.htm>
- <http://www.javaworld.com/columns/w-Java-design-patterns-index.shtml>
- <http://www.onjava.com/pub/a/onjava/2002/01/16/patterns.html?page=1>
- <http://www.core2eepatterns.com/index.htm>
- <http://www.theserverside.com/books/wiley/EJBDesignPatterns/index.tss>
- <http://www.martinfowler.com/eaaCatalog/>

**Q 12:** How would you go about determining the enterprise security requirements for your Java/J2EE application?

**A 12:** It really pays to understand basic security terminology and J2EE security features. Let's look at them:

Some of the key security concepts are:

- Authentication
- Authorisation ((J2EE declarative & programmatic)
- Data Integrity
- Confidentiality and privacy
- Non-repudiation and auditing

<b>Terminology</b>	<b>Description</b>
Authentication	Authentication is basically an identification process. <b>Do I know who you are?</b>

	<p><b>Terminology used for J2EE security:</b></p> <p><b>Principal:</b> An entity that can be identified and authenticated. For example an initiator of the request like user etc).</p> <p><b>Principal name:</b> Identity of a principal like user id etc.</p> <p><b>Credential:</b> Information like password or certificate, which can authenticate a principal.</p> <p><b>Subject:</b> A set of principals and their credentials associated with a thread of execution.</p> <p><b>Authentication:</b> The process by which a server verifies the identity presented by a user through username/userid and password or certificate etc. For example the username and password supplied by the user can be validated against an LDAP server or a database server to verify he is whom he claims to be.</p> <p><b>Authentication methods:</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Basic/Digest authentication:</b> Browser specific and password is encoded using Base-64 encoding. Digest is similar to basic but protects the password through encryption. This is a simple <b>challenge-response</b> scheme where the client is challenged for a user id and password. The Internet is divided into <b>realms</b>. A realm is supposed to have one user repository so a combination of user id and password is unique to that realm. The user challenge has the name of the realm so that users with different user ids and password on different systems know which one to apply. Lets look at a HTTP user challenge format           <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <b>WWW-Authenticate: Basic realm="<i>realm_name</i>"</b> </div>           The user-agent (ie Web browser) returns the following HTTP header field           <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <b>Authorization: Basic <i>userid:password</i></b> </div>           With <b>Basic authentication</b> the user id and password string is base64 encoded. The purpose of base64 is to avoid sending possibly unprintable or control characters over an interface that expects text characters. It does not provide any security because the clear text can be readily restored.         </li> <li><input type="checkbox"/> <b>Digest authentication</b>: The server challenges the user with a "nonce" which is an unencrypted random value. The user responds with a <b>checksum</b> (typically MD5 hash) of the user id, password, the nonce and some other data. The server creates the same checksum from the user parameters available in the user registry. If both the checksums match then it is assumed that the user knows the correct password.</li> <li><input type="checkbox"/> <b>Form-based authentication</b>: Most Web applications use the form-based authentication since it allows applications to customise the authentication interface. Uses base64 encoding which can expose username and password unless all connections are over SSL. (Since this is the most common let us look at in greater detail together ie authentication &amp; authorisation under Authorisation).</li> <li><input type="checkbox"/> <b>Certificate based authentication</b>: Uses PKI and SSL. This is by far the most secured authentication method. A user must provide x509 certificate to authenticate with the server.</li> </ul>
Authorisation	<p>Authorization is the process by which a program determines whether a given identity is permitted to access a resource such as a file or an application component. <b>Now that you are authenticated, I know who you are? But Are you allowed to access the resource or component you are requesting?</b></p> <p><b>Terminology used for J2EE security:</b></p> <p><b>Authorization:</b> Process of determining what type of access (if any) the security policy allows to a resource by a principal.</p> <p><b>Security role:</b> A logical grouping of users who share a level of access permissions.</p> <p><b>Security domain:</b> A scope that defines where a set of security policies are maintained and enforced. Also known as security policy domain or realm.</p> <p>J2EE uses the concept of <b>security roles</b> for both declarative and programmatic access controls. This is different from the traditional model, which is <b>permission-based</b> (for example UNIX file system security where resources like files are associated with a user or group who might have permission to read the file but not execute).</p> <p>Let us look at some differences between</p> <p><b>Permission-based authorization:</b> Typically the <b>permission-based</b> security both users and resources are defined in a security database and the association of user and groups with the resources takes place through <b>Access Control Lists (ACL)</b>. The maintenance of these registry and ACLs requires a security</p>

	<p>administrator.</p> <p><b>Role based authorization:</b> In J2EE role based model, the users and groups of users are still stored in a user registry. A mapping must also be provided between <b>users and groups</b> to the <b>security constraints</b>. This can exist in a registry or <b>J2EE applications themselves have their own role based security constraints</b> defined through deployment descriptors like web.xml, ejb-jar.xml, and/or application.xml. So the applications themselves do not have to be defined by a security administrator.</p> <p>Now lets look at <b>role based authorization</b> in a bit more detail:</p> <p>J2EE has both a <b>declarative and programmatic way of protecting individual method of each component</b> (Web or EJB) by specifying which security role can execute it.</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Refer <b>Q23</b> in Enterprise section.</li> <li><input type="checkbox"/> Refer <b>Q81</b> in Enterprise section</li> <li><input type="checkbox"/> Also refer <b>Q7</b> in Enterprise section for the <i>deployment descriptors</i> where <b>&lt;security-role&gt;</b> are defined.</li> </ul> <p>Let's look at the commonly used form-based authentication and authorisation in a bit more detail.</p> <p><b>STEP:1</b> The <b>web.xml</b> defines the type of authentication mechanism</p> <pre>&lt;login-config&gt; &lt;auth-method&gt;FORM&lt;/auth-method&gt; &lt;realm-name&gt;FBA&lt;/realm-name&gt; &lt;form-login-config&gt; &lt;form-login-page&gt;myLogon&lt;/form-login-page&gt; &lt;form-error-page&gt;myError&lt;/form-error-page&gt; &lt;/form-login-config&gt; &lt;/login-config&gt;</pre> <p><b>STEP: 2</b> The user creates a form that must contain fields for username, password etc as shown below. The names should be as shown for fields in bold:</p> <pre>&lt;form method="POST" action="<b>j_security_check</b>"&gt; &lt;input type="text" name="<b>j_username</b>"&gt; &lt;input type="text" name="<b>j_password</b>"&gt; &lt;/form&gt;</pre> <p><b>STEP: 3</b> Set up a security realm to be used by the container. Since LDAP or database provide flexibility and ease of maintenance, Web containers have support for different types of security realms like LDAP, Database etc.</p> <p>For example Tomcat Web container uses the server.xml to set up the database as the security realm.</p> <pre>&lt;realm classname="org.apache.catalina.realm.JDBCRealm" debug="99" drivername="org.gjt.mm.mysql.Driver" connectionurl="jdbc:mysql://localhost/tomcatusers?user=test;password=test" usertable="users" <b>usernamecol="user_name" usercredcol="user_pass"</b> <b>userroletable="user_roles" rolenamecol="role_name"/&gt;</b></pre> <p>You have to create necessary tables and columns created in the database.</p> <p><b>STEP: 4</b> Set up the security constraints in the web.xml for authorisation.</p> <pre>&lt;security-constraint&gt; &lt;web-resource-collection&gt; &lt;web-resource-name&gt;PrivateAndSensitive&lt;/web-resource-name&gt; &lt;url-pattern&gt;/private/*&lt;/url-pattern&gt; &lt;/web-resource-collection&gt; &lt;auth-constraint&gt; &lt;role-name&gt;executive&lt;/role-name&gt; &lt;role-name&gt;admin&lt;/role-name&gt; &lt;/auth-constraint&gt; &lt;/security-constraint&gt;</pre> <p><b>The Web containers perform the following steps to implement security when a protected Web resource is accessed:</b></p> <p><b>Step 1:</b> Determine whether the user has been authenticated.</p> <p><b>Step 2:</b> If the user has not been already authenticated, request the user to provide security credentials by redirecting the user to the login page defined in the web.xml as per Step-1 &amp; Step-2 described above.</p>
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p><b>Step 3:</b> Validate the user credentials against the <b>security realm</b> set up for the container.</p> <p><b>Step 4:</b> Determine whether the authenticated user is <b>authorised</b> to access the Web resource defined in the deployment descriptor web.xml. Web containers enforce authorization on a page level. For fine grained control programmatic security can be employed using</p> <pre>request.getRemoteUser(), request.isUserInRole(), request.getUserPrincipal() etc</pre> <p>Note: The Web containers can also propagate the authentication information to EJB containers.</p>
Data integrity	<p>Data integrity helps to make sure if something is <b>intact and not tampered with</b> during transmission.</p> <p><b>Checksums:</b> Simply add up the bytes within file or request message. If the checksums match the integrity is maintained. The weakness with this approach is that some times junks can be added to make sums equal like</p> <pre>ABCDE == EDCBA</pre> <p><b>Cryptography hashes:</b> This uses a mathematical function to create small result called <b>message digest</b> from the input message. Difficult to create false positives. Common hash functions are <b>MD5, SHA</b> etc.</p> <p><b>Data</b> [e.g. Name is Peter] → <b>MD5 iterative hash function</b> → <b>Digest</b> [e.g. f31d120d3]</p> <p>It is not possible to change the message digest back to its original data. You can only compare two message digests i.e. one came with the client's message and the other is recomputed by the server from sent message. If both the message digests are equal then the message is intact and has not been tampered with.</p>
Confidentiality and Privacy	<p>The confidentiality and privacy can be accomplished through encryption. Encryption can be:</p> <p><b>Symmetric or private-key:</b> This is based on a single key. This requires the sender and the receiver to share the same key. Both must have the key. The sender encrypts his message with a private key and the receiver decrypts the message with his own private key. This system is not suitable for large number of users because it requires a key for every pair of individuals who need to communicate privately. As the number of participants increases then number of private keys required also increases. So a company which wants to talk to 1000 of its customers should have 1000 private keys. Also the private keys need to be transmitted to all the participants, which has the vulnerability to theft. <b>The advantages of the symmetric encryption are its computational efficiency and its security.</b></p> <p><b>Asymmetric or public-key infrastructure (PKI):</b> This is based on a pair of mathematically related keys. One is a public key, which is distributed to all the users, and the other key is a private key, which is kept secretly on the server. So this requires only two keys to talk to 1000 customers. This is also called Asymmetric encryption because <b>the message encrypted by public key can only be decrypted by the private key</b> and the message encrypted by the private key can only be decrypted by the public key.</p> <p>In a public key encryption anybody can create a key pair and publish the public key. So we need to verify the owner of the public key is who you think it is. So the creator of this false public key can intercept the messages intended for some one else and decrypt it. To protect this public key systems provide mechanisms for validating the public keys using <b>digital signatures</b> and <b>digital certificates</b>.</p> <p><b>Digital signature:</b> A digital signature is a stamp on the data, which is unique and very difficult to forge. A <b>digital signature has 2 steps</b> and establishes 2 things from the security perspective.</p> <p><b>STEP 1:</b> To sign a document means hashing software (e.g. MD5, SHA) will crunch the data into just a few lines by the process called '<b>hashing</b>'. These few lines are called <b>message digest</b>. <u>It is not possible to change the message digest back to its original data.</u> Same as what we saw above in cryptography hashes. <b>This establishes whether the message has been modified between the time it was digitally signed and sent and time it was received by the recipient.</b></p> <p><b>STEP 2:</b> Computing the digest can verify the integrity of the message but does not stop from some one intercepting it or <b>verifying the identity of the signer</b>. This is where encryption comes into picture. Signing the message with the private key will be useful for proving that the message must have come from the user who claims to have signed it. <b>The second step in creating a digital signature involves encrypting the digest code created in STEP 1 with the sender's private key.</b></p> <p>When the message is received by the recipient the following steps take place:</p> <ol style="list-style-type: none"> <li>1. Recipient recomputes the digest code for the message.</li> <li>2. Recipient decrypts the signature by using the sender's public key. This will yield the original digest code of the sender.</li> <li>3. Compare the original and the recomputed digest codes. If they match then the message is both intact and signed by the user who claims to have signed it (i.e. authentic).</li> </ol>

	<p><b>Digital Certificates:</b> A certificate represents an organisation in an official digital form. This is equivalent to an electronic identity card which serves the purpose of</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Identifying the owner of the certificate. This is done with authenticating the owner through trusted 3rd parties called the certificate authorities (CA) e.g. Verisign etc. The CA digitally signs these certificates. When the user presents the certificate the recipient validates it by using the digital signature.</li> <li><input type="checkbox"/> Distributing the owner's public key to his users (or recipients of the message).</li> </ul> <p>The <b>server certificates</b> let visitors to your website exchange personal information like credit card number etc with the server with the confidence that they are communicating with intended site and not the rogue site impersonating the intended site. Server certificates are must for e-commerce sites. <b>Personal certificates</b> let you authenticate a visitor's identity and restrict access to specified content to particular visitors. Personal certificates are ideal for business-to-business communication where offering partners and suppliers special access to your website.</p> <p>A certificate includes details about the owner of the certificate and the issuing CA. A certificate includes:</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Distinguished name (DN) of the owner, which is a unique identifier. You need the following for the DN: <ul style="list-style-type: none"> <li>• Country Name (C)</li> <li>• State (ST)</li> <li>• Locality (L)</li> <li>• Organization Name (O)</li> <li>• Organization Unit (OU)</li> <li>• Common Name (CN)</li> <li>• Email Address.</li> </ul> </li> <li><input type="checkbox"/> Public key of the owner.</li> <li><input type="checkbox"/> The issue date of the certificate.</li> <li><input type="checkbox"/> The expiry date of the certificate.</li> <li><input type="checkbox"/> The distinguished name of the issuing CA.</li> <li><input type="checkbox"/> The digital signature of the issuing CA.</li> </ul> <p>Now lets look at the core concept of the certificates:</p> <p><b>STEP 1:</b> The owner makes a request to the CA by submitting a certificate request with the above mentioned details. The certificate request can be generated with tool like OpenSSL REQ, Java keytool etc. This creates a certreq.perm file, which can be transferred to CA via FTP.</p> <pre>-----BEGIN NEW CERTIFICATE REQUEST----- MIIBJTCB0AIBADBtMQswCQYDVQQGEwJVUzEQMA4GA1UEChs4IBMHQXJpem9uYTEN A1UEBxMETWVzYTEfMB0GA1UEChMWTWVs3XbnzYSBDb21tdW5pdHkgQ29sbGVnZTE A1UEAxMTd3d3Lm1jLm1hcmljb3BhLmVkdTBaMA0GCSqGSIb3DQEBAQUAA0kAMEYC QQDRNU6xslWjG41163gArsj/P108sFmjkjzMuUUFYbtZX4RFxf/U7cZZdMagz4I MmY0F9cdpDLTAutULTsZKDcLAgEDoAAwDQYJKoZlhcNAQEEBQADQQAjIfpTLgfm BVhc9Sqaip5SFNXtzAmhYzvJkt5JJ4X2r7VJYG3J0vauJ5VkjXz9aevJ8dzx37ir 3P4XpZ+NFxK1R= -----END NEW CERTIFICATE REQUEST-----</pre> <p><b>STEP 2:</b> The CA takes the owner's certificate request and creates a message 'm' from the request and signs the message 'm' with CA's private key to create a separate signature 'sig'. The message 'm' and the signature 'sig' form the certificate, which get sent to the owner.</p> <p><b>STEP 3:</b> The owner then distributes both parts of the certificate (message and signature) to his customers (or recipients) after signing the certificate with owner's private key.</p> <p><b>STEP 4:</b> The recipient of the certificate (i.e. the customer) extracts the certificate with owner's public key and subsequently verifies the signature 'sig' using CA's public-key. If the signature proves valid, then the recipient accepts the public key in the certificate as the owner's key.</p>
Non-repudiation and auditing	Proof that the sender actually sent the message. It also prohibits the author of the message from falsely denying that he sent the message. This is achieved by record keeping the exact time of the message transmission, the public key used to decrypt the message, and the encrypted message itself. Record keeping can be complicated but critical for non-repudiation.
Secure Socket Layer (SSL)	Secure Socket Layer (SSL) uses a combination of symmetric and asymmetric (public-key) encryption to accomplish confidentiality, integrity, authentication and non-repudiation for Internet communication. In a nutshell SSL uses public key encryption to confidentially transmit a session key which can be used to conduct symmetric encryption. SSL uses the public key technology to negotiate a shared session key between the client and the server. The public key is stored in an X.509 certificate that usually has a digital signature from a trusted 3 <sup>rd</sup> party like Verisign. Lets look at the handshake sequence where the server and the client negotiate the cipher suite to be used, establish a shared session key and authenticate server to

	the client and optionally client to the server.
	<ul style="list-style-type: none"> <li><input type="checkbox"/> Client requests a document from a secure server <a href="https://www.myapp.com.au">https://www.myapp.com.au</a>.</li> <li><input type="checkbox"/> The server sends its X.509 certificate to the client with its public key stored in the certificate.</li> <li><input type="checkbox"/> The client checks whether the certificate has been issued by a CA it trusts.</li> <li><input type="checkbox"/> The client compares the information in the certificate with the site's public key and domain name.</li> <li><input type="checkbox"/> Client tells the server what cipher suites it has available.</li> <li><input type="checkbox"/> The server picks the strongest mutually available ciphers suite and notifies the client.</li> <li><input type="checkbox"/> The client generates a <b>session key</b> (symmetric key or private key) and encrypts it using the server's public key and sends it to the server.</li> <li><input type="checkbox"/> The server receives the encrypted session key and decrypts it using its private key.</li> <li><input type="checkbox"/> The client and server use the session key to encrypt and decrypt the data they send to each other.</li> </ul>

**Note:** Use HTTP **post** as opposed to HTTP **get** (sends sensitive information as a query string appended to your URL) in your web based applications, since it is more secured due to hiding sensitive information from your URL query string. Your URL query string can be easily tampered with to determine any security holes in your application.

**Q 13:** How would you go about describing the open source projects like JUnit (unit testing), Ant (build tool), CVS (version control system) and log4J (logging tool) which are integral part of most Java/J2EE projects?

**A 13:** JUnit, ANT and CVS are integral part of most Java/J2EE projects. JUnit for unit testing, ANT for deployment, and CVS for source control. Let's look at each, one by one. I will be covering only the key concepts, which can be used as a reference guide in addition to being handy in interviews.

### JUnit

This is a regression testing framework, which is used by developers who write unit tests in Java. **Unit testing** is relatively inexpensive and easy way to produce better code faster. Unit testing exercises testing of a very small specific functionality. To run JUnit you should have JUnit.jar in your classpath.

**Unix:** CLASSPATH=\$CLASSPATH:/usr/Java/packages/junit3.8.1/JUnit.jar

**Dos:** CLASSPATH=%CLASSPATH%;C:\junit3.8.1\JUnit.jar

JUnit can be coded to run in two different modes as shown below:

Per test mode	Per suite setup (more common)
<p>The per test mode will call the <code>setUp()</code> method before executing every test case and <code>tearDown()</code> method after executing every test case.</p> <p>Lets look at an example:</p> <pre>import junit.framework.TestCase; public class SampleTest extends TestCase {     Object o = null;     protected void setUp() {         System.out.println("running setUp()");         //Any database access code         //Any set up code         o = new Object();     }     protected void tearDown() {         System.out.println("running tearDown()");         //Any clean up code         o = null;     }     public void testCustomer() {         System.out.println("running testCustomer()");         assertEquals(5, 2 + 3);     }     public void testAccount() {         System.out.println("running testAccount()");     } }</pre> <p>public void testCustomer() {      System.out.println("running testCustomer()");      assertEquals(5, 2 + 3);      assertNotEqual("check if it is null", o);      assertTrue(5 == 5);  }</p> <p>public void testAccount() {      System.out.println("running testAccount()");  }</p>	<p>In this mode the <code>setUp()</code> and <code>tearDown()</code> will be executed only once:</p> <pre>import junit.framework.*; import junit.extensions.*;  public class SampleTest2 extends TestCase {     Object o = null;     public void testCustomer() {         System.out.println("running testCustomer()");         assertEquals(5, 2 + 3);     }     public void testAccount() {         System.out.println("running testAccount()");     }     public SampleTest2(String method) {         super(method);     }     public static Test suite() {         TestSuite suite = new TestSuite();         suite.addTest(new SampleTest2("testCustomer"));         suite.addTest(new SampleTest2("testAccount"));         TestSetup wrapper = new TestSetup(suite) {             protected void setUp() {</pre>

```

System.out.println("running testAccount()");
if (5 < 3)
 fail();
}

}

```

as per the above example the execution sequence is as follows:

```

running setUp()
running testAccount()
running tearDown()
running setUp()
running testCustomer()
running tearDown()

```

```

 oneTimeSetUp();
 }

 protected void tearDown() {
 oneTimeTearDown();
 }

};

return wrapper;
}

public static void oneTimeSetUp() {
 System.out.println("running setUp()");
 //runs only once to setup
}
public static void oneTimeTearDown() {
 System.out.println("running tearDown()");
 //runs only once to cleanup
}

```

as per the above example the execution sequence is as follows:

```

running setUp()
running testCustomer()
running testAccount()
running tearDown()

```

## How to run JUnit?

**Text mode:** java -cp <junit.jar path> junit.textui.TestRunner

**Graphics mode:** java -cp <junit.jar path> junit.swingui.TestRunner

The smallest groupings of test expressions are the **methods** that you put them in. Whether you use JUnit or not, you need to put your test expressions into Java methods, so you might as well group the expressions, according to any criteria you want, into methods. An object that you can run with the JUnit infrastructure is a **Test**. But you can't just implement Test and run that object. You can only *run* specially created instances of **TestCase**. A **TestSuite** is just an object that contains an ordered list of runnable Test objects. TestSuites also implement Test() and are runnable. **TestRunners** execute Tests, TestSuites and TestCases.

## ANT (Another Niche Tool)

Ant is a tool which helps you build, test, and deploy (Java or other) applications. ANT is a command-line program that uses a XML file (i.e. build.xml) to describe the build process. The build.xml file describes the various tasks ant has to complete. ANT is a very powerful, portable, flexible and easy to use tool. Ant has the following command syntax:

**ant [ant-options] [target 1] [target 2] [...target n]**

**Some ant options are:**

-help, -h	: print list of available ant-options (ie prints this message)
-verbose	: be extra verbose
-quiet	: be extra quiet
-projecthelp , -p	: print project help information
-buildfile <file>	: use given build file
-logger <classname>	: class which is to perform logging
-D<property=value>	: use value for given property
-propertyfile <name>	: load all properties from file with -D properties taking precedence.
-keep-going, -k	: execute all targets that do not depend on failed targets
... and more	

Let's look at a simple build.xml file:

```

?xml version="1.0" encoding="UTF-8"?
<project name="MyProject" default="compile" basedir=".">
<property name="src" value=".\\src" />

```

```

<property name="build" value=".\\classes\" />

<target name="init">
<mkdir dir="${build}" />
</target>

<target name="compile" description="compiles the packages" depends="init">
<javac srcdir="${src}" destdir="${build}" optimize="on" debug="on">
 <classpath>
 <pathelment location="${build}" />
 </classpath>
</javac>
</target>

<target name="clean" description="cleans the build directory">
<delete dir="${build}" />
</target>

</project>

```

We can run the above with one of the following commands

```

$ ant compile
$ ant clean compile
$ ant -b build.xml compile

```

Now lets look at some of the key concepts:

Concept	Explanation with example
Ant Targets	<p>An Ant build file contains one <b>project</b>, which itself contains multiple <b>targets</b>. Each target contains <b>tasks</b>. Targets can depend on each other, so building one target may cause others to be built first.</p> <p>From the above build.xml file example</p> <p><b>name:</b> Name of the target to run.</p> <p><b>description:</b> A target determines whether the target defined as internal or public based on description. If the description attribute is defined then it is public and otherwise it is internal. In the above example targets compile and clean are public. The target init is internal. When you run the following command option the public targets are displayed.</p> <pre>ant -projecthelp</pre> <p><b>depends:</b> The target “compile” depends on the target “init”. So the target init will be run before target compile is run.</p> <p><b>If:</b> If the given property has been defined then the target will be executed.</p> <pre>&lt;target name="A" if="somePropertyName1"&gt;   &lt;echo message="I am in target A"&gt; &lt;/target&gt;</pre> <p><b>unless:</b> If the given property is <b>not defined</b> then the target will be executed.</p> <pre>&lt;target name="B" unless="somePropertyName2"&gt;   &lt;echo message="I am in target B"&gt; &lt;/target&gt;</pre> <p>Ant delegates work to other targets as follows:</p> <pre>&lt;target name="build" depends="prepare"&gt;   &lt;antcall target="compile" /&gt;   &lt;antcall target="jar" /&gt; &lt;/target&gt;</pre>
Ant tasks	<p>Ant task is where real work is done. A task can take any number of attributes. Ant tasks can be categorised as follows:</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> <b>Core tasks:</b> Tasks that are shipped with core distribution like &lt;javac ...&gt;, &lt;jar ...&gt; etc</li> <li><input type="checkbox"/> <b>Optional tasks:</b> Tasks that require additional jar files to be executed like &lt;ftp ....&gt; etc</li> <li><input type="checkbox"/> <b>User defined tasks:</b> Tasks that are to be developed by users by extending Ant framework.</li> </ul>

	<p>For example &lt;javac&gt; is a task.</p> <pre>&lt;javac srcdir="\${src}" destdir="\${build}" optimize="on" debug="on"&gt;   &lt;classpath&gt;     &lt;pathelement location="\${build}" /&gt;   &lt;/classpath&gt; &lt;/javac&gt;</pre>
Ant data types	<p>Ant data types are different to the ones in other programming languages. Lets look at some of the ant data types.</p> <p><b>description:</b></p> <pre>&lt;project default="deploy" basedir="."&gt;   &lt;description&gt; This is my project&lt;/description&gt; &lt;/project&gt;</pre> <p><b>patternset:</b></p> <ul style="list-style-type: none"> <li>? → matches a single character</li> <li>* → matches 0 or more characters</li> <li>** → matches 0 or more directory recursively</li> </ul> <pre>&lt;patternset id="classfile"&gt;   &lt;include name="**/*.class" /&gt;   &lt;exclude name="**/*Test*.class" /&gt; &lt;/patternset&gt;</pre> <p><b>dirset:</b></p> <pre>&lt;dirset dir="\${build.dir}"&gt;   &lt;patternset id="classfile"&gt;     &lt;include name="**/classes" /&gt;     &lt;exclude name="**/*Test*" /&gt;   &lt;/patternset&gt; &lt;/dirset&gt;</pre> <p><b>fileset:</b></p> <pre>&lt;fileset dir="\${build.dir}"&gt;   &lt;include name="**/*.Java" /&gt;   &lt;exclude name="**/*Test*" /&gt; &lt;/fileset&gt;</pre> <p><b>filelist, filer, filterchain, filterreader, selectors, xmlicatalogs etc</b></p>
Lets look at some key tasks where:	<p><b>Fetch code updates from CVS:</b></p> <pre>&lt;target name="cvsupdate" depends="prepare"&gt;   &lt;cvsroot cvsroot="\${CVSROOT}" passwd="\${rep.passwd}" /&gt;   &lt;cvs cvsRoot="\${CVSROOT}" command="update -p -d" failOnError="true" /&gt; &lt;/target&gt;</pre> <p><b>Run unit tests with JUnit:</b></p> <pre>&lt;target name="test" depends="compile"&gt;   &lt;junit failureproperty="\${testsFailed}" &gt;     &lt;classpath&gt;       &lt;pathelement path="\${classpath}" /&gt;       &lt;pathelement path="\${build.dir.class}" /&gt;     &lt;/classpath&gt;     &lt;formatter type="xml"/&gt;     &lt;test name="mytests.testall" todir="\${reports.dir}" /&gt;   &lt;/junit&gt; &lt;/target&gt;</pre> <p><b>Creating a jar file:</b></p> <pre>&lt;target name="jar" depends="test" unless="testsFailed"&gt;   &lt;jar destfile="\${build.dir}/\${name}.jar" basedir="\${build.dir}" include="**/*.class" /&gt; &lt;/target&gt;</pre>

	<p><b>Email the results with the help of loggers:</b></p> <p>Now let's look at how we can e-mail the run results. Ant has <b>listeners</b> and <b>loggers</b>. A listener is a component that is alerted to certain events during the life of a build. A logger is built on top of the listener and is the component that is responsible for logging details about the build. The listeners are alerted to 7 different events like build started, build finished, target started, target finished, task started, task finished and message logged.</p> <p>The loggers are more useful and interesting. You are always using a logger when you run ant (i.e. DefaultLogger). You can specify the logger as shown below:</p> <pre>ant -logger org.apache.tools.ant.listener.MailLogger</pre> <p>You can also specify other loggers like XmlLogger, Log4Jlistener etc.</p> <p>The MailLogger logs whatever information comes its way and then sends e-mail. A group of properties must be set for a MailLogger which can be passed on to ant as a standard command-line Java option &lt;ie -DmailLogger.mailhost="blah.com" &gt; or the &lt;property ...&gt; statements in the init target. Lets look at some of the properties to be set:</p> <pre>MailLogger.mailhost MailLogger.from MailLogger.failure.notify → whether to send an e-mail on build failure. MailLogger.success.notify → whether to send an e-mail on build success. MailLogger.fail.to MailLogger.success.to</pre>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Note:** **Maven** is a software project management and comprehension tool, which is gaining popularity. Maven is based on the concept of project object model (**POM**), and it can manage a project's build process, reporting and documentation from a centralized piece of information. Maven provides a uniform build system where by requiring a single set of Ant build files that can be shared by all projects using Maven. Maven provides following information about your project: Change logs from your repository information, cross referenced sources, source metrics, mailing lists, developer lists, dependency lists, unit test reports including coverage etc.

## CVS

CVS is a version control or tracking system. It maintains records of files through their development and allows retrieval of any stored version of a file, and supports production of multiple versions.

### cvs [cvs-options] command [command-options-arguments]

CVS allows you to split the development into 2 or more parts called a trunk (MAIN) and a branch. You can create 1 or more branches. Typically a branch is used for bug fixes and trunk is used for future development. Both the trunk and branches are stored in the same repository. This allows the change from branch (i.e. bug fixes) to ultimately or periodically be merged into the main trunk ensuring that all bug fixes get rolled into next release.

Unlike some other version control systems, CVS instead of locking files to prevent conflicts (i.e. when 2 developers modifying the same file) it simply allows multiple developers to work on the same file. Subsequently with the aid of cvs file merging feature it allows you to merge all the changes into one file. The benefits of version control systems like CVS include:

- Any stored revision of a file can be retrieved to be viewed or changed.
- Differences between 2 revisions can be displayed.
- Patches can be created automatically.
- Multiple developers can simultaneously work on the same file.
- Project can be branched into multiple streams for varied tasks and then branches can be merged back into trunk (aka **MAIN**).
- Also supports distributed development and can be configured to record commit messages into a bug tracking system.

Let's look at some of the key concepts and commands.

Concept	Explanation with examples
Building a repository	<p>The repository should be built on a partition that is backed up and won't shut down. The repositories are stored under '<b>cvsroot</b>' i.e. /var/lib/cvsroot or /cvsroot. The command to set up the chosen directory as a CVS repository:</p> <pre>cvs -d /var/lib/cvsroot</pre>

	Lets look at some command line examples:  <pre>\$ mkdir /var/lib/cvsroot \$ chgrp team /var/lib/cvsroot \$ chmod g+srwx /var/lib/cvsroot \$ cvs -d /var/lib/cvsroot</pre>
Importing projects	After creating a repository you can import a project or a related collection of files stored under a single directory by using the following command:  <pre>cvs [-d &lt;repository-path&gt;] import &lt;project_name&gt; &lt;vendor_tag&gt; &lt;related_tag&gt;</pre> Lets look at some command line examples:  <pre>\$ cd /tmp \$ mkdir ProjectX \$ touch ProjectX/File1.Java \$ touch ProjectX/File2.Java \$ touch ProjectX/File3.Java \$ cd ProjectX \$ cvs -d /var/lib/cvsroot import ProjectX INITIAL start</pre>
Creating a <b>sandbox</b> , checking out and updating files from cvs repository into a <b>sandbox</b>	Copy of the files, which gets checked out by the client from the cvs repository, is called a <b>sandbox</b> . The user can manipulate the files within the sandbox and when the files have been modified they can be resubmitted into the repository with the changes. Lets look at how to create a sandbox (i.e. a client working copy):  <pre>cvs -d /var/lib/cvsroot checkout ProjectX</pre> The above command will result in creating a subdirectory called ProjectX under the present working directory.  Subsequently to keep the sandbox in sync with the repository, an update command can be executed. The update command checks your checked-out cvs sandbox against the cvs repository and down loads any changed files into the sandbox from the repository.  <pre>cvs update -d</pre>
Adding files into cvs repository from a sandbox.	To add file from sandbox into cvs repository you should create a file first.  <pre>\$ touch file3 \$ cvs add file3 \$ cvs commit</pre> To add directories and files  <pre>\$ cvs add design plan design/*.rtf plan/*.rtf</pre>
Checking file stats and help	<pre>\$ cvs [cvs-options] stats [command-option] &lt;filename&gt; \$ cvs -help \$ cvs rlog ProjectX</pre>
Removing a file from the cvs repository.	To remove a file from the repository, first remove the file from the sandbox directory and then run the cvs command.  <pre>\$ rm file3 \$ cvs remove file3 \$ cvs commit</pre>
Moving or renaming files	To move or rename files:  <pre>\$ mv file1 file101 \$ cvs remove file1 \$ cvs add file101 \$ cvs commit</pre>
Releasing a sandbox	CVS release should be used before deleting a sandbox. CVS first checks whether there are any files with uncommitted changes.

	\$ cvs release
Tagging files	<p>Tagging is a way of marking a group of file revisions as belong together. If you want to look at all the file revisions belonging to a tag the cvs will use the tag string to locate all the files.</p> <p>To tag files in the repository</p> <pre>\$ cvs -d /var/lib/cvsroot rtag -r HEAD release_1 ProjectX</pre> <p>To tag files in the sandbox</p> <pre>\$ cvs tag release_1</pre>
Removing tags	<p>To remove a tag from sandbox.</p> <pre>\$ cvs tag -d release_1 file1</pre> <p>To remove a tag from repository.</p> <pre>\$ cvs rtag -d release_1 file1</pre>
Retrieving files based on past revisions instead of the latest files.	<p>We have already looked at how to checkout latest code. What if we want to checkout by a revision?</p> <pre>\$ cvs checkout -r Tagname ProjectX</pre> <p>To update by revision</p> <pre>\$ cvs update -d -r release_1</pre>
Creating branches	<p>Branches can be added to the repository tree in order to allow different development paths to be tried, or to add parallel development of code to different base versions.</p> <p>To create a branch from sandbox, you can use</p> <pre>\$ cvs update -d -r release_1 \$ cvs tag -r release_1 -b release_1_branch</pre> <p>To create a branch from the repository</p> <pre>\$ cvs rtag -r release_1 -b release_1_branch</pre> <p>As shown in the diagram it is always a good practice to tag the trunk at the root of branch before branching. This makes it easier to merge the changes back to trunk later. It is also a</p>

	<p>good practice to tag the branch at the root of the branch prior to merging back to head.</p> <p>To merge from branch to trunk (HEAD)</p> <pre>cvs update -j branch_base_tag -j branchname</pre> <pre>\$ cvs update -j release_1 -j release_1_branch</pre> <p>To make subsequent merges from the branch to trunk(HEAD)</p> <pre>\$ cvs update -j release_1_branch_merge_1 -j release_1_branch</pre> <p>To merge from trunk to branch</p> <pre>\$ cvs update -j release_1_branch_merge_1 -j HEAD</pre>
CVS admin task	<p>To add binary files like images, documents etc to cvs</p> <pre>\$ cvs add -kb image.jpg</pre> <pre>\$ cvs add -kb acceptance.doc</pre>

#### Log4J

Refer **Q126** in Enterprise section.

**Q 14:** How would you go about describing Web services?

**A 14:** This book would not be complete without mentioning Web services. Web services provide application-to-application communication using XML.

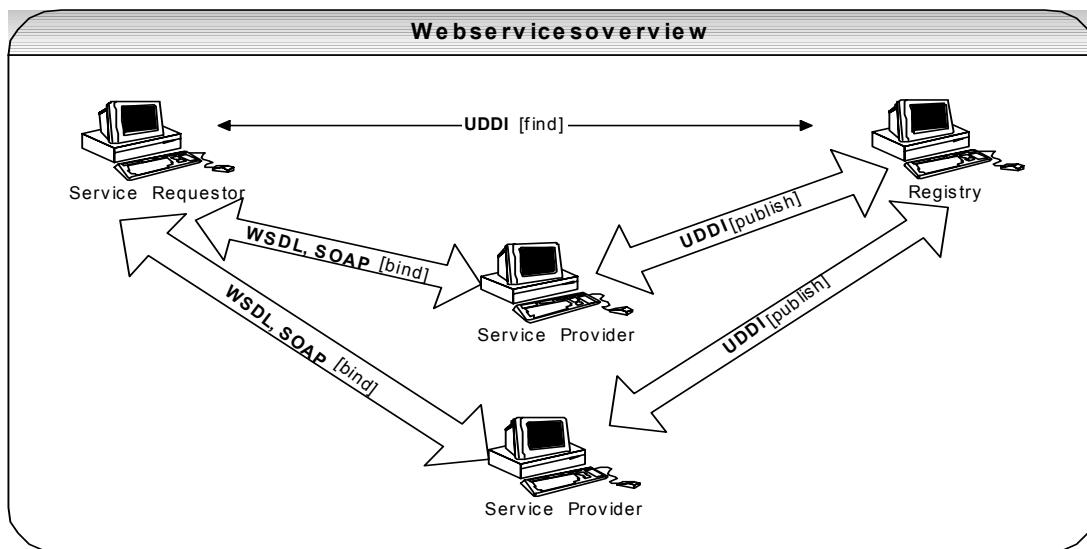
**What is the difference between a Web (website) and a Web service?**

Web (website)	Web Service
<p>A <b>Web</b> is a scalable information space with interconnected resources. A Web interconnects resources like Web pages, images, an application, word document, e-mail etc.</p> <p><b>service-oriented architecture (SOA)</b> is a design pattern in which application components provide services to other components via a communications protocol, typically over a network.</p> <p>In Data Service layer - DOTNET is planning to reuse the logic. So I can implement DAO layer as a service. This is called SOA.</p>	<p>A <b>Web service</b> is a <b>service</b>, which lives on the <b>Web</b>. A Web service posses both the characteristics of a <b>Web</b> and a <b>service</b>. We know what a Web is; let's look at what a service is?</p> <p>A service is an application that exposes its functionality through an API (Application Programming Interface). So what is a component you may ask? A <b>service</b> is a <b>component</b> that can be used remotely through a remote interface either synchronously or asynchronously. The term service also implies something special about the application design, which is called a <b>service-oriented architecture (SOA)</b>. One of the most important features of SOA is the <b>separation of interface from implementation</b>. A service exposes its functionality through interface and interface hides the inner workings of the implementation. The client application (ie user of the service) only needs to know how to use the interface. The client does not have to understand actually how the service does its work. For <b>example</b>: There are so many different models of cars like MAZDA, HONDA, TOYOTA etc using different types of engines, motors etc but as a user or driver of the car you do not have to be concerned about the internals. You only need to know how to start the car, use the steering wheel etc, which is the interface to you.</p> <p>Usually a service runs on a server, waiting for the client application to call it and ask to do some work? These services are often run on application servers which manage scalability, availability, reliability, multi-threading, transactions, security etc.</p>
Designed to be consumed by humans (ie users, clients, business partners etc). For <b>example</b> : <a href="http://www.google.com">www.google.com</a> is a Web search engine that contains index to more than 8 billion of other Web pages. The normal interface is Web browser like Internet Explorer, which is used by human.	<p>Designed to be consumed by software (i.e. other applications).</p> <p>For <b>example</b>: Google also provides a Web service interface through the Google API to query their search engine from an application rather than a browser. Refer <a href="http://www.google.com/apis/">http://www.google.com/apis/</a> for Google Web API</p>
	<p><b>Principles of SOA</b></p> <ol style="list-style-type: none"> <li>1 Standard service contract between provider and consumer</li> <li>2 Loose coupling to reduce dependency and maintain only awareness of each other</li> <li>3 Abstraction to hide logic from outside world</li> <li>4 Autonomous so that services alone have total control over the logic</li> </ol> <ul style="list-style-type: none"> <li>* Reusable</li> <li>* Statelessness</li> <li>* Granularity: it is simple and small</li> <li>* Interoperability</li> </ul>

- 1 Standard service contract between provider and consumer
  - 2 Loose coupling to reduce dependency and maintain only awareness of each other
  - 3 Abstraction to hide logic from outside world
  - 4 Autonomous so that services alone have total control over the logic
- \* Reusable
  - \* Statelessness
  - \* Granularity: it is simple and small
  - \* Interoperability

**Why use Web services when you can use traditional style middleware such as RPC, CORBA, RMI and DCOM?**

Traditional middleware	Web Services
Tightly coupled connections to the application and it can break if you make any modification to your application. Tightly coupled applications are hard to maintain and less reusable.	Web Services support loosely coupled connections. The interface of the Web service provides a layer of abstraction between the client and the server. The loosely coupled applications reduce the cost of maintenance and increases reusability.
Generally does not support heterogeneity.	Web Services present a new form of middleware based on XML and Web. <b>Web services are language and platform independent.</b> You can develop a Web service using any language and deploy it on to any platform, from small device to the largest supercomputer. <b>Web service uses language neutral protocols such as HTTP and communicates between disparate applications by passing XML messages</b> to each other via a Web API.
Does not work across Internet.	Does work across Internet.
More expensive and hard to use.	Less expensive and easier to use.


**Let's look at some of the key terms**

Terms	Explanation
XML	XML provides the way to structure data and XML provides the foundation on which Web services are built.
SOAP	<b>SOAP</b> stands for <b>Simple Object Access Protocol</b> . It is an XML based lightweight protocol, which allows software components and application components to communicate, mostly using HTTP. SOAP sits on top of the HTTP protocol. <b>SOAP is nothing but XML message based document with pre-defined format.</b> SOAP is designed to communicate via the Internet in a platform and language neutral manner and allows you to get around firewalls as well. Lets look at some SOAP messages:

- A SOAP message MUST be encoded using XML
- A SOAP message MUST use the SOAP Envelope namespace
- A SOAP message MUST use the SOAP Encoding namespace
- A SOAP message must NOT contain a DTD reference
- A SOAP message must NOT contain XML Processing Instructions

```

<?xml version="1.0"?>
<soap:Envelope
 xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
 soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
 <soap:Header>
 ...
 ...
 </soap:Header>
 <soap:Body>
 ...
 ...
 </soap:Body>
 <soap:Fault>
 ...
 </soap:Fault>

```

```

...
</soap:Fault>
</soap:Body>
</soap:Envelope>
```

Let's look at a SOAP request and a SOAP response:

#### **SOAP Request:**

```

POST /Price HTTP/1.1
Host: www.mysite.com
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 300

<?xml version="1.0"?>
<soap:Envelope
 xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
 soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
 <soap:Body>
 <m:GetPrice xmlns:m="http://www.mysite.com/prices">
 <m:Item>PlasmaTV</m:Item>
 </m:GetPrice>
 </soap:Body>
</soap:Envelope>
```

#### **SOAP Response:**

```

HTTP/1.1 200 OK
Content-Type: application/soap; charset=utf-8
Content-Length: 200

<?xml version="1.0"?>
<soap:Envelope
 xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
 soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
 <soap:Body>
 <m:GetPriceResponse
 xmlns:m="http://www.mysite.com/prices">
 <m:Price>3500.00</m:Price>
 </m:GetPriceResponse>
 </soap:Body>
</soap:Envelope>
```

Lets look at a HTTP header:

```

POST /Price HTTP/1.1
Host: www.mysite.com
Content-Type: text/plain
Content-Length: 200
```

#### **SOAP HTTP Binding**

A SOAP method is an HTTP request/response that complies with the SOAP encoding rules.

**HTTP + XML = SOAP**

Let's look at a HTTP header containing a soap message:

```

POST /Price HTTP/1.1
Host: www.mysite.com
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 200
```

	<p><b>WSDL</b> (Web Services Description Language) WSDL stands for <b>Web Service Description Language</b>. A WSDL document is an XML document that describes how the messages are exchanged. Let's say we have created a Web service. Who is going to use that and how does the client know which method to invoke and what parameters to pass? There are tools that can generate WSDL from the Web service. Also there are tools that can read a WSDL document and create the necessary code to invoke the Web service. So the WSDL is the Interface Definition Language (IDL) for Web services.</p>
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p><b>UDDI</b> (Universal Description Discovery and Integration) UDDI provides a way to publish and discover information about Web services. UDDI is like a registry rather than a repository. A registry contains only</p>
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

and Integration)	<p>reference information like the JNDI, which stores the EJB stub references. UDDI has white pages, yellow pages and green pages. If the retail industry published a UDDI for a price check standard then all the retailers can register their services into this UDDI directory. Shoppers will search the UDDI directory to find the retailer interface. Once the interface is found then the shoppers can communicate with the services immediately.</p> <p>The Web services can be registered for public use at <a href="http://www.uddi.org">http://www.uddi.org</a>. Once the Web service is selected through the UDDI then it can be located using the discovery process.</p> <p>Before UDDI, there was no Internet standard for businesses to reach their customers and partners with information about their products and services. Neither was there a method of how to integrate businesses into each other's systems and processes. UDDI uses WSDL to describe interfaces to Web services</p>
------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Next section covers some of the popular emerging technologies & frameworks. Some organizations might be considering or already started using these technologies. All these have emerged over the past 3 years. So it is vital that you have at least a basic understanding of these new paradigms and frameworks because these new paradigms and frameworks can offer great benefits such as ease of maintenance, reduction in code size, elimination of duplication of code, ease of unit testing, loose coupling among components, light weight and fine grained objects etc.**

## SECTION FOUR

### Emerging Technologies/Frameworks...

This section covers some of the popular emerging technologies you need to be at least aware of, if you have not already used them. If there are two or more interview candidates with similar skills and experience then awareness or experience with some of the emerging technologies can play a role in the decision. Some organizations might be considering or already started using these technologies. So it is well worth your effort to demonstrate that you understand the basics of technologies like:

- Test Driven Development (**TDD**).
- Aspect Oriented Programming (**AOP**).
- Inversion of Control (**IOC**) (Also known as **Dependency Injection**).
- Annotation or attribute based programming (xdoclet etc).
- Spring framework.
- Hibernate framework.
- EJB 3.0
- JavaServer Faces (**JSF**)

### Why should you seriously consider these technologies?

This section covers some of the new and popular design paradigms such as Plain Old Java Objects (POJOs) and Plain Old Java Interfaces (POJI) based services and interceptors, Aspect Oriented Programming (AOP), Dependency Injection (DI), and tools and frameworks, which apply these new paradigms such as Spring, Hibernate, EJB 3.0, XDoclet, JSF, etc. All these have emerged over the past 3 years. These new paradigms and frameworks can offer great benefits such as ease of maintenance, reduction in code size, elimination of duplication of code, ease of unit testing, loose coupling among components, light weight and fine grained objects etc.

**Q 01:** What is Test Driven Development (TDD)?

**A 01:** TDD is an iterative software development process where you **first write the test with the idea that it must fail**. This is a different approach to the traditional development where you write the application functionality first and then write test cases. The major benefit of this approach is that the code becomes thoroughly unit tested (you can use **JUnit** or other unit testing frameworks). For JUnit refer **Q13** on "How would you go about..." section. TDD is based on two important principles preached by its originator Kent Beck:

- **Write new business code only if an automated unit test has failed:** Business application requirements drive the tests and tests drive the actual functional code. Each test should test only one business concept, which means avoid writing a single test which tests withdrawing money from an account and depositing money into an account. Any change in the business requirements will impact pre and post conditions of the test. Talking about pre and post conditions, following **design by contract** methodology (Refer **Q9** in Java section) helps achieving TDD. In design by contract, you specify the pre and post conditions that act as contracts of a method, which provides specification to write your tests against.
- **Eliminate duplication from the code:** A particular business concept should be implemented only once within the application code. A code for checking an account balance should be centralized to only one place within the application code. This makes your code decoupled, more maintainable and reusable.

I can hear some of you all saying how can we write the unit test code without the actual application code. Let's look at how it works in steps. The following steps are applied iteratively for business requirements.

**STEP: 1** write some tests for a specific business requirement.

**STEP: 2** write some basic structural code so that your **test compiles but the test should fail** (failures are the pillars of success). **For example** just create the necessary classes and corresponding methods with skeletal code.

**STEP: 3** write the required business code to pass the tests which you wrote in step 1.

**STEP: 4** finally refactor the code you just wrote to make it is as simple as it can be. You can refactor your code with confidence that if it breaks the business logic then you have unit test cases that can quickly detect it.

**STEP: 5** run your tests to make sure that your refactored code still passes the tests.

**STEP: 6** Repeat steps 1-5 for another business requirement.

To write tests efficiently some basic guidelines need to be followed:

- You should be able to run each test in isolation and in any order.
- The test code should not have any duplicate business logic.
- You should test for all the pre and post conditions as well as exceptions.
- Each test should concentrate on one business requirement as mentioned earlier.
- There are many ways to write test conditions so proper care and attention should be taken. In some cases pair programming can help by allowing two brains to work in collaboration. You should have strategies to overcome issues around state of data in RDBMS (Should you persist sample test data, which is a snapshot of your actual data prior to running your tests? Or should you hard code data? Or Should you combine both strategies? Etc).

**Q 02:** What is the point of Test Driven Development (TDD)?

**A 02:** TDD process improves your confidence in the delivered code for the following reasons.

- TDD can eliminate duplication of code and also disciplines the developer to focus his mind on delivering what is absolutely necessary. This means the system you develop only does what it is suppose to do

(because you first write test cases for the business requirements and then write the required functionality to satisfy the test cases) and no more.

- These unit tests can be repeatedly run to alert the development team immediately if some one breaks any existing functionality. All the unit tests can be run overnight as part of deployment process and test results can be emailed to the development team.
  - TDD ensures that code becomes thoroughly unit tested. It is not possible to write thorough unit tests if you leave it to the end due to deadline pressures, lack of motivation etc.
  - TDD complements design by contract methodology and gets the developer thinking in terms of pre and post conditions as well as exceptions.
  - When using TDD, tests drive your code and to some extent they assist you in validating your design at an earlier stage.
  - TDD also helps you refactor your code with confidence that if it breaks the business logic it gets picked up when you run your unit tests next time.
  - TDD promotes **design to interface not implementation** design concept. **For example:** when your code has to take input from an external source or device which is not present at the time of writing your unit tests, you need to create an interface, which takes input from another source in order for your tests to work.
- 

**Q 03:** What is aspect oriented programming? Explain AOP?

**A 03:** Aspect-Oriented Programming (**AOP**) complements OOP (Object Oriented Programming) by allowing the developer to dynamically modify the static OO model to create a system that can grow to meet new requirements.

AOP allows you to dynamically modify your static model consisting mainly of business logic to include the code required to fulfil the secondary requirements or in AOP terminology called **cross-cutting concerns** (secondary requirements) like auditing, logging, security, exception handling etc without having to modify the original static model (in fact, we don't even need to have the original code). Better still, we can often keep this additional code in a single location rather than having to scatter it across the existing model, as we would have to if we were using OOP on its own.

**For example:** A typical Web application will require a servlet to bind the HTTP request to an object and then passes to the business handler object to be processed and finally return the response back to the user. So only a minimum amount of code is initially required. But once you start adding all the other additional secondary requirements or cross-cutting concerns like logging, auditing, security, exception-handling etc the code will inflate to 2-4 times its original size. This is where AOP can assist by separately modularizing these cross-cutting concerns and integrating these concerns at runtime or compile time through aspect weaving. AOP allows rapid development of evolutionary prototype using OOP by focussing only on the business logic by omitting concerns such as security, auditing, logging etc. Once the prototype is accepted, additional concerns like security, logging, auditing etc can be woven into the prototype code to transfer it into a production standard application.

AOP nomenclature is different from OOP and can be described as shown below:

**Join points:** represents the point at which a cross-cutting concern like logging, auditing etc intersects with a main concern like the core business logic. Join points are locations in programs' execution path like method & constructor call, method & constructor execution, field access, class & object initialization, exception handling execution etc.

**pointcut:** is a language construct that identifies specific join points within the program. A pointcut defines a collection of join points and also provides a context for the join point.

**Advice:** is an implementation of a cross-cutting concern which is a piece of code that is executed upon reaching a pointcut within a program.

**Aspect:** encapsulates join points, pointcuts and advice into a reusable module for the cross-cutting concerns which is equivalent to Java classes for the core concerns in OOP. Classes and aspects are independent of one another. **Classes are unaware of the presence of aspects, which is an important AOP concept.** Only pointcut declaration binds classes and aspects.

**Weaving** is the process for interleaving separate cross-cutting concerns such as logging into core concern such as business logic code to complete the system. AOP weaver composes different implementations of aspects into a cohesive system based on weaving rules. The weaving process (aka injection of aspects into Java classes) can happen at:

- **Compile-time:** Weaving occurs during compilation process.
- **Load-time:** Weaving occurs at the byte-code level at class loading time.
- **Runtime:** Similar to load-time where weaving occurs at byte-code level during runtime as join points are reached in the executing application.

**So which approach to use?** Load-time and runtime weaving have the advantages of being highly dynamic and enabling changes on the fly without having to rebuild and redeploy. But Load-time and runtime weaving adversely affect system performance. Compile time weaving offers better performance but requires rebuilding and redeployment to effect changes.

Let's look at AOP language constructs with code samples. At a higher level AOP language has two parts:

- **The AOP language specification:** describes language constructs and syntax. The specification has two high level steps:

**STEP 1:** Implementation of individual aspects like business logic, logging, security etc into corresponding individual code so that a compiler can translate it into executable code. You can use languages like Java to implement individual code.

Let's look at a basic code for business logic aspect:

```
public class AccountProcessor {
 public void deposit(Currency amount) {
 //depositing logic only, no log statements
 }

 public void withdraw(Currency amount) throws InsufficientFundsException {
 //withdrawing logic only, no log statements
 }
}
```

Let's look at a separate basic code for logging aspect

```
public interface Logger {
 public void log (String message);
}
```

**STEP 2:** Code weaving rules for individually composed code in step 1 to form a final system. This is achieved through specifying the weaving rules through a language, which is responsible for composing individual code developed in step 1. The weaving rule in our example is:

- Log public business logic method deposit() at the beginning.

The language for specifying weaving rules could be an extension of the implementation language. Let's look at a sample specification code for weaving written in *AspectJ* (freely available AOP for Java from Xerox PARC):

```
/** aspect for logging */

public aspect LogAccountProcessor Operations {
 Logger logger = new ConsoleLogger();

 /** join point declaration */
 pointcut logAccount(): call (* AccountProcessor.deposit(..));

 /** advice to execute before call */
 before() : logAccount() {

 logger.log("Amount=" + amount);
 }
}
```

- **AOP language implementation:** verifies the code's correctness according to the AOP specification and converts it into byte code that target JVM can execute. AOP language compilers perform two logical steps:

**STEP 1:** Combine the individual aspects based on the weaving rules.

**STEP 2:** Convert the resulting information into executable byte code.

For Java based AOP implementation, the JVM would load the weaving rules and then apply those rules to subsequently loaded aspect classes by performing just-in-time aspect weaving.

The AOP generated resulting code after applying the weaving rules will look something like:

```
public class AccountProcessorWithLogging {
 Logger _logger;

 public void deposit(Currency amount) {
 _logger.log("Amount=" + amount); // joint point defined
 //depositing logic
 }

 public void withdraw(Currency amount) throws InsufficientFundsException {
 //withdrawing logic only, no log statements because no join point defined.
 }
}
```

**Q 04:** What are the differences between OOP and AOP?

**A 04:**

Object Oriented Programming (OOP)	Aspect Oriented Programming (AOP)
OOP looks at an application as a set of collaborating objects. OOP code scatters system level code like logging, security etc with the business logic code.	AOP looks at the complex software system as combined implementation of multiple concerns like business logic, data persistence, logging, security, multithread safety, error handling, and so on. Separates business logic code from the system level code. In fact one concern remains unaware of other concerns.
OOP nomenclature has classes, objects, interfaces etc.	AOP nomenclature has join points, point cuts, advice, and aspects.
Provides benefits such as code reuse, flexibility, improved maintainability, modular architecture, reduced development time etc with the help of polymorphism, inheritance and encapsulation.	AOP implementation coexists with the OOP by choosing OOP as the base language. <b>For example:</b> AspectJ uses Java as the base language.  AOP provides benefits provided by OOP plus some additional benefits which are discussed in the next question.

**Q 05:** What are the benefits of AOP?

**A 05:**

- OOP can cause the system level code like logging, transaction, security etc to scatter throughout the business logic. AOP helps overcome this problem by centralising the cross-cutting concerns.
- AOP addresses each aspect separately in a modular fashion with minimal coupling and duplication of code. This modular approach also promotes code reuse by using a business logic concern with a separate logger aspect.
- It is also easier to add newer functionalities by adding new aspects and weaving rules and subsequently regenerating the final code. This ability to add newer functionality as separate aspects enable application designers to delay or defer some design decisions without the dilemma of over designing the application.
- Promotes rapid development of evolutionary prototype using OOP by focussing only on the business logic by omitting cross-cutting concerns such as security, auditing, logging etc. Once the prototype is accepted, additional concerns like security, logging, auditing etc can be weaved into the prototype code to transfer it into a production standard application.

- Developers can concentrate on one aspect at a time rather than having to think simultaneously about business logic, security, logging, performance, multithread safety etc. Different aspects can be developed by different developers based on their key strengths. **For example:** A security aspect can be developed by a security expert or a senior developer who understands security.

**Q 06:** What is attribute or annotation oriented programming?

**A 06:** Before looking at attribute oriented programming let's look at code generation processes. There are two kinds of code generation processes.

**Passive code generation:** is template driven. Input wizards are used in modern IDEs like eclipse, Websphere Studio Application Developer (WSAD) etc where parameters are supplied and the code generator carries out the process of parameter substitution and source code generation. **For example:** in WSAD or eclipse you can create a new class by supplying the “New Java class” wizard appropriate input parameters like class name, package name, modifiers, superclass name, interface name etc to generate the source code. Another example would be *Velocity* template engine, which is a powerful Java based generation tool from the Apache Software Foundation.

**Active code generation:** Unlike passive code generators the active code generators can inject code directly into the application as and when required.

**Attribute/Annotation oriented programming languages** leverages the active code generation with the use of declarative tags embedded within the application source code to generate any other kind of source code, configuration files, deployment descriptors etc. These declarative metadata tags are called **attributes** or **annotations**. The purpose of these **attributes** is to extend the functionality of the base language like Java, with the help of custom attributes provided by other providers like Hibernate framework, Spring framework, XDoclet etc. The attributes or annotations are specified with the symbol “@<label>”. JDK1.5 has a built-in runtime support for attributes.

Let's look at an example. Say we have a container managed entity bean called Account. Using attribute oriented programming we can generate the deployment descriptor file ejb-jar.xml by embedding some attributes within the bean implementation code.

```
/*
 * @ejb.bean
 * name="Account"
 * jndi-name ="ejb/Account"
 */
public abstract class AccountBean implements EntityBean {
 ...
}
```

The above-embedded attributes can generate the ejb-jar.xml as shown below using XDoclet:

```
<ejb-jar>
<entity>
<ejb-name>Account</ejb-name>
<home>com.AccountHome</home>
<remote>com.Account</remote>
<ejb-class>com.AccountBean</ejb-class>
...
</entity>
</ejb-jar>
```

**Q 07:** What are the pros and cons of annotations over XML based deployment descriptors?

**A 07:** Service related attributes in your application can be configured through a XML based deployment descriptor files or annotations. XML based deployment descriptor files are processed separately from the code, often at runtime, while annotations are compiled with your source code and checked by the compiler.

XML	Annotations
More verbose because has to duplicate a lot of information like class names and method names from your code.	Less verbose since class names and method names are part of your code.
Less robust due to duplication of information which	More robust because annotations are processed with your code

introduces multiple points for failure. If you misspell a method name then the application will fail.	and checked by the compiler for any discrepancies and inaccuracies.
More flexible since processed separately from the code. Since it is not hard-coded can be changed later. Your deployment team has a great flexibility to inspect and modify the configuration.	<p>Less flexible since annotations are embedded in Java comment style within your code.</p> <p>For example, to define a stateless session EJB 3.0 with annotations, which can serve both local and remote clients:</p> <pre><b>@Stateless</b> <b>@Local</b> ({LocalCounter.class}) <b>@Remote</b> ({RemoteCounter.class})  public class CounterBean implements LocalCounter, RemoteCounter { ... }</pre>
XML files can express complex relationships and hierarchical structures at the expense of being verbose.	Annotations can hold only a small amount of configuration information and most of plumbing has to be done in the framework.

**Which one to use?** Annotations are suitable for most application needs. XML files are more complex and can be used to address more advanced issues. XML files can be used to override default annotation values. Annotations cannot be used if you do not have access to source-code. The decision to go with annotation or XML depends upon the architecture behind the framework. For example Spring is primarily based on XML and EJB 3.0 is primarily based on annotations, but both support annotations and XML to some degree. EJB 3.0 uses XML configuration files as an optional overriding mechanism and Spring uses annotations to configure some Spring services.

**Q 08:** What is XDoclet?

**A 08:** XDoclet is an open source code generation engine for **attribute oriented programming** from SourceForge.net (<http://xdoclet.sourceforge.net/xdoclet/index.html>). So you add attributes (i.e. metadata) in JavaDoc style tags (@ejb.bean) and XDoclet will parse your source files and JavaDoc style attributes provided in the Java comment with @ symbol to generate required artifacts like XML based deployment descriptors, EJB interfaces etc. XDoclet can generate all the artifacts of an EJB component, such as remote & local interfaces as well as deployment descriptors. You place the required attributes on the relevant classes and methods that you want to process.

**Q 09:** What is inversion of control (IOC) (also known as dependency injection)?

**A 09:** Inversion of control or dependency injection is a term used to resolve component dependencies by injecting an instantiated component to satisfy dependency as opposed to explicitly requesting a component. So components will not be explicitly requested but components are provided as needed with the help of an Inversion of controller containers. This is analogous to the Hollywood principal where the servicing components say to the requesting client code "don't call us, we'll call you". Hence it is called inversion of control.

Most of you all are familiar with the software development context where client code (requesting code) collaborates with other dependent components (or servicing components) by knowing which components to talk to, where to locate them and how to talk with them. This is achieved by embedding the code required for locating and instantiating the requested components within the client code. The above approach will tightly couple the dependent components with the client code. This tight coupling can be resolved by applying the **factory design pattern** and **program to interfaces not to implementations driven development**. But the factory design pattern is still an intrusive mechanism because servicing components need to be requested explicitly. Let us look at how dependency injection comes to our rescue. It takes the approach that clients declare their dependency on servicing components through a configuration file (like xml) and some external piece of code assumes the responsibility of locating and instantiating these servicing components and supplying the relevant references when needed to the client code. This external piece of code is often referred to as IOC (aka dependency injection) container or framework.

IOC or dependency injection containers generally control creation of objects (by calling "new") and resolve dependencies between objects it manages. Spring framework, Pico containers, Hivemind etc are IoC containers to name a few. IOC containers support **eager instantiation**, which is quite useful if you want self-starting services that "come up" on their own when the server starts. They also support **lazy loading**, which is useful when you have many services which may only be sparsely used. Here is pseudo code for how IOC would work:

XML declaration (beans.xml) showing objects that need to be instantiated and dependencies between them. Dependency is declared as property element.

```
<beans>
 <bean id="FlightReservation" class="com.FlightReservation" />
 <bean id="FlightReservation" class="com.HotelReservation" />
 <bean id="TripPlanner" class="com.TripPlanner">
 <property name="flight"><ref bean="FlightReservation"/></property>
 <property name="hotel"><ref bean="HotelReservation"/></property>
 </bean>

</beans>
```

To initialize the container

```
ClassPathResource res = new ClassPathResource("beans.xml");
BeanFactory factory = new XmlBeanFactory(res);
```

The references to the implementation can be retrieved based on "id" attribute in the xml configuration file and all the dependent components are implicitly instantiated in specified order and setter methods (i.e. **setter injection**) are called to resolve the dependencies.

```
factory.getBean("TripPlanner");
```

Dependencies can be wired by either using Java code or using XML.

**Q 10:** What are the different types of dependency injections?

**A 10:** There are three types of dependency injections.

- Constructor Injection (e.g. Pico container, Spring etc): Injection is done through constructors.
- Setter Injection (e.g. Spring): Injection is done through setter methods.
- Interface Injection (e.g. Avalon): Injection is done through an interface.

**Q 11:** What are the benefits of IOC (aka Dependency Injection)?

**A 11:**

- Minimises the amount of code in your application. With IOC containers you do not care about how services are created and how you get references to the ones you need. You can also easily add additional services by adding a new constructor or a setter method with little or no extra configuration.
- Make your application more testable by not requiring any singletons or JNDI lookup mechanisms in your unit test cases. IOC containers make unit testing and switching implementations very easy by manually allowing you to inject your own objects into the object under test.
- Loose coupling is promoted with minimal effort and least intrusive mechanism. The factory design pattern is more intrusive because components or services need to be requested explicitly whereas in IOC the dependency is injected into requesting piece of code. Also some containers promote the design to interfaces not to implementations design concept by encouraging managed objects to implement a well-defined service interface of your own.
- IOC containers support eager instantiation and lazy loading of services. Containers also provide support for instantiation of managed objects, cyclical dependencies, life cycles management, and dependency resolution between managed objects etc.

**Q 12:** What is the difference between a service locator pattern and an inversion of control pattern?

**A 12:**

Service locator	Inversion of Control (IOC)
The calling class which needs the dependent classes needs to tell the service locator which classes are needed. Also the calling classes have the responsibility of finding these dependent classes and invoking them. This makes the classes tightly coupled with each other.	In IoC (aka Dependency Injection) pattern the responsibility is shifted to the IoC containers to locate and load the dependent classes based on the information provided in the descriptor files. Changes can be made to the dependent classes by simply modifying the descriptor files. This approach makes the dependent classes loosely coupled with the calling class.

Difficult to unit test the classes separately due to tight coupling.	Easy to unit test the classes separately due to loose coupling.
----------------------------------------------------------------------	-----------------------------------------------------------------

**Q 13:** Why dependency injection is more elegant than a JNDI lookup to decouple client and the service?

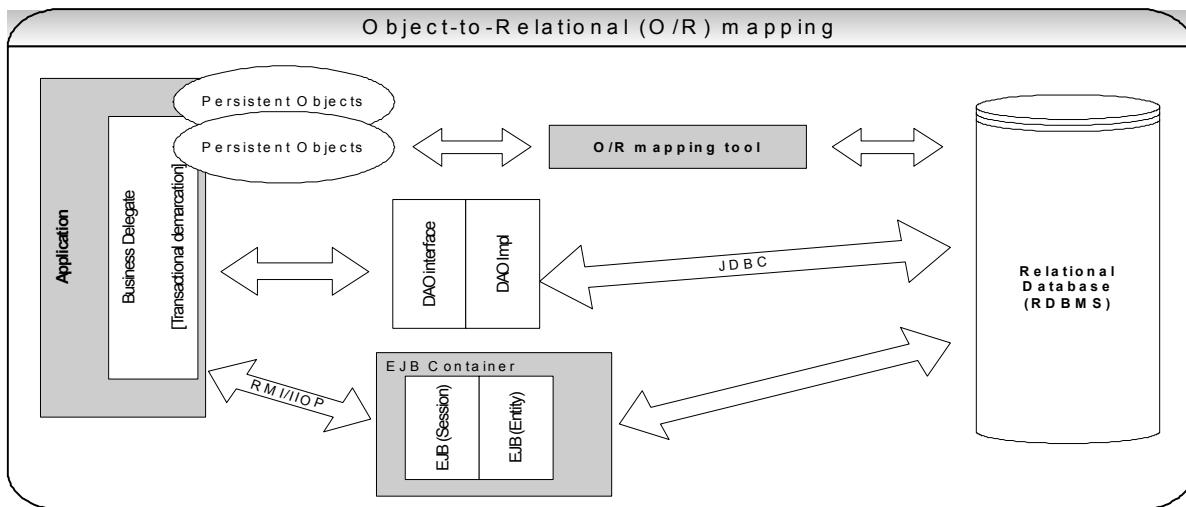
**A 13:** Here are a few reasons why a JNDI look up is not elegant:

- The client and the service being looked up must agree on a string based name, which is a contract not enforced by the compiler or any deployment-time checks. You will have to wait till runtime to discover any discrepancies in the string based name between the lookup code and the JNDI registry.
- The JNDI lookup code is verbose with its own try-catch block, which is repeated across the application.
- The retrieved service objects are not type checked at compile-time and could result in casting error at runtime.

Dependency injection is more elegant because it promotes loose coupling with minimal effort and least intrusive mechanism. Dependency is injected into requesting piece of code by the IOC containers like Spring etc. With IOC containers you do not care about how services are created and how you get references to the ones you need. You can also easily add additional services by adding a new constructor or a setter method with little or extra configuration.

**Q 14:** Explain Object-to-Relational (O/R) mapping?

**A 14:** There are several ways to persist data and the persistence layer is one of the most important layers in any application development. O/R mapping is a technique of mapping data representation from an object model to a SQL based relational model.



O/R mapping is well suited for read → modify → write centric applications and not suited for write centric applications (i.e. batch processes with large data sets like 5000 rows or more) where data is seldom read. O/R mapping tools allow you to model inheritance (Refer **Q101** in Enterprise section), association and composition class relationships. O/R mapping tools work well in 80-90% of cases and use most of the basic database features like stored procedures, triggers etc, when O/R mapping is not appropriate. Keep in mind that no one size fits all solution. Always validate your architectural design with a vertical slice and test for performance. Some times you have to handcraft your SQL and a good O/R mapping tool should allow that. O/R mapping tools allow your application to be less verbose, more portable and more maintainable.

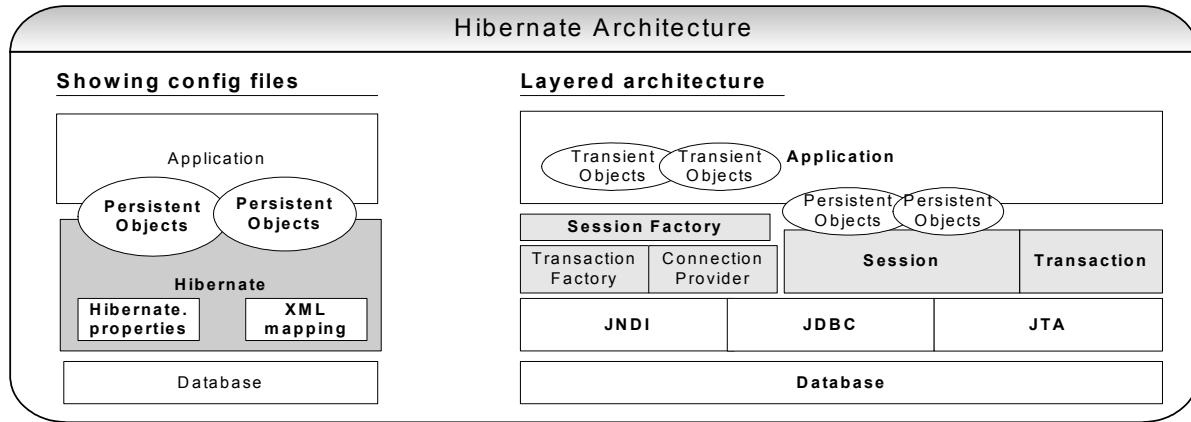
**Q 15:** Give an overview of hibernate framework?

**A 15:** Hibernate is a full-featured, open source Object-to-Relational (O/R) mapping framework. Unlike EJB, Hibernate can work inside or outside of a J2EE container. Hibernate works with Plain Old Java Objects (POJOs), which is much like a JavaBean.

**SessionFactory** is Hibernate's concept of a single datastore and is threadsafe so that many threads can access it concurrently and request for sessions and immutable cache of compiled mappings for a single database. A

SessionFactory is usually only built once at startup with a load-on-startup servlet. SessionFactory should be wrapped in some kind of singleton so that it can be easily accessed in an application code.

```
SessionFactory sf = new Configuration().configure().buildSessionFactory();
```



**Session** is a non-threadsafe object that represents a single unit-of-work with the database. Sessions are opened by a SessionFactory and then are closed when all work is complete. To avoid creating too many sessions ThreadLocal class can be used as shown below to get the current session no matter how many times you make call to the `getCurrentSession()` method.

```
...
public class HibernateUtil {
 ...
 public static final ThreadLocal local = new ThreadLocal();
 ...
 public static Session currentSession() throws HibernateException {
 Session session = (Session) local.get();
 //open a new session if this thread has no session
 if(s == null) {
 session = sf.openSession();
 local.set(session);
 }
 return session;
 }
}
```

It is also vital that you close your session after your unit of work completes.

**Transaction** is a single threaded, short lived object used by the application to specify atomicity. Transaction abstracts your application code from underlying JDBC, JTA or CORBA transaction. At times a session can span several transactions. A **TransactionFactory** is a factory for transaction instances. A **ConnectionProvider** is a factory for pool of JDBC connections. A ConnectionProvider abstracts an application from underlying Datasource or DriverManager.

```
Transaction tx = session.beginTransaction();
Employee emp = new Employee();
emp.setName("Brian");
emp.setSalary(1000.00);

session.save(emp);
tx.commit();
//close session
```

**Persistent Objects and Collections** are short lived single threaded objects, which store the persistent state and some business functions. They are Plain Old Java Objects (POJOs) and are currently associated with one session. As soon as the associated session is closed, persistent objects are detached and free to use directly as transient data transfer objects in any application layers like business layer, presentation layer etc.

**Transient – object is instantiated but not associated with session**

**Persistent – object is associated with session**

**Detached – object exists but detached from session (session is closed)**

**Transient Object and Collections** are instances of persistent objects that are currently not associated with a session. They can be either instantiated by an application, which has not yet been persisted or they may have been instantiated by a closed session.

### Configuring Hibernate

Hibernate mappings can be either configured manually through a simple readable XML format (\*.hbm.xml) or by embedding mapping information directly in the source code using Hibernate doclet (@hibernate.<tags>), which is a sibling to XDoclet and automatically generating the mapping XML file using an Ant script.

### Querying with Hibernate

Hibernate provides very robust querying API that supports query strings, named queries and queries built as aggregate expressions. The most flexible way is using the Hibernate Query Language syntax (HQL), which is easy to understand and is an Object Oriented extension to SQL, which supports inheritance and polymorphism.

```
Query query = session.createQuery("Select emp from Employee as emp where emp.sex = :sex");
query.setCharacter("sex",'F');
```

Type-safe queries can be handled by object oriented query by criteria approach. You can also use Hibernate's direct SQL query feature. If none of the above meets your requirements then you can get a plain JDBC connection from a Hibernate session.

**Q 16:** Explain some of the pitfalls of Hibernate and explain how to avoid them?

**A 16:**

- Use the ThreadLocal session pattern when obtaining Hibernate session objects (Refer **Q15** in Emerging Technologies/Frameworks). This is important because Hibernate's native API does not use the current thread to maintain associations between session and transaction or between session and application thread.
- Handle resources properly by making sure you properly flush and commit each session object when persisting information and also make sure you release or close the session object when you are finished working with it. Most developers fall into this pitfall. If you pass a connection object to your session object then remember to issue **session.close().close()** which will first release the connection back to the pool and then will close the session. If you do not pass a connection object then issue **session.close()** to close the session.
- Use **lazy associations** when you use relationships otherwise you can unwittingly fall into the trap of executing unnecessary SQL statements in your Hibernate applications. Let us look at an example: Suppose we have a class *Employee* with many-to-one relationship with class *Department*. So one department can have many employees. Suppose we want to list the name of the employees then we will construct the query as follows:

```
Query query = session.createQuery("from Employee emp");
List list = query.list();
```

Hibernate will generate the following SQL query:

```
SELECT <fields> from Employee;
```

If it only generates the query above then it is okay and it serves our purpose, but we get another set of SQL queries without asking it to do anything. One for each of the referenced departments in *Department* table. If you had 5 departments then the following query will be executed 5 times with corresponding department id. This is the N+1 selects problem. In our example it is 5 + 1. Employee table is queried once and Department table is queried 5 times.

```
SELECT <fields> from Department where DEPARTMENT.id=?
```

**Solution** is to make the *Department* class lazy, simply by enabling the lazy attribute in the *Department*'s hbm.xml mapping definition file, which will result in executing only the first statement from the *Employee* table and not the 5 queries from the *Department* table.

```
<class name="com.Department" table="Department" lazy="true" > </class>
```

Only one query is required to return an employee object with its department initialized. In Hibernate, lazy loading of persistent objects are facilitated by proxies (i.e. virtual proxy design pattern). In the above example you have a *Department* object, which has a collection of *Employee* objects. Let's say that *Employee* objects are lazy loaded. If you make a call *department.getEmployees()* then Hibernate will load only the employeeIDs

and the version numbers of the *Employee* objects, thus saving loading of individual objects until later. So what you really have is a collection of proxies not the real objects. The reason being, if you have hundreds of employees for a particular department then chances are good that you will only deal with only a few of them. So, why unnecessarily instantiate all the *Employee* objects? This can be big performance in some situations.

- **Avoid N+1 Selects problem:** Having looked at the N+1 problem occurring inadvertently due to not having a lazy association in the previous example, now what if we need the Departmental information in addition to the Employee details. It is not a good idea to execute N+1 times.

```
<class name="com.Department" table="Department" lazy="true" > </class>
```

Now to retrieve *Departmental* info you would:

```
Query query = session.createQuery("from Employee emp");
List list = query.list();
Iterator it = list.iterator();

while(it.hasNext()) {
 Employee emp = (Employee) it.next();
 emp.getDepartment().getName(); //N+1 problem. Since Department is not already loaded so
 //additional query is required for each department.
}
```

The solution is to make sure that the initial query retrieves all the data needed to load the objects by issuing a HQL fetch join (**eager loading**) as shown below:

```
"from Employee emp join fetch emp.Department dept"
```

The above HQL results in an inner join SQL as shown below:

```
SELECT <fields from Employee & Department> FROM employee
inner join department on employee.departmentId = department.id.
```

Alternatively you can use a criteria query as follows:

```
Criteria crit = session.createCriteria(Employee.class);
crit.setFetchMode("department", FetchMode.EAGER);
```

The above approach creates the following SQL:

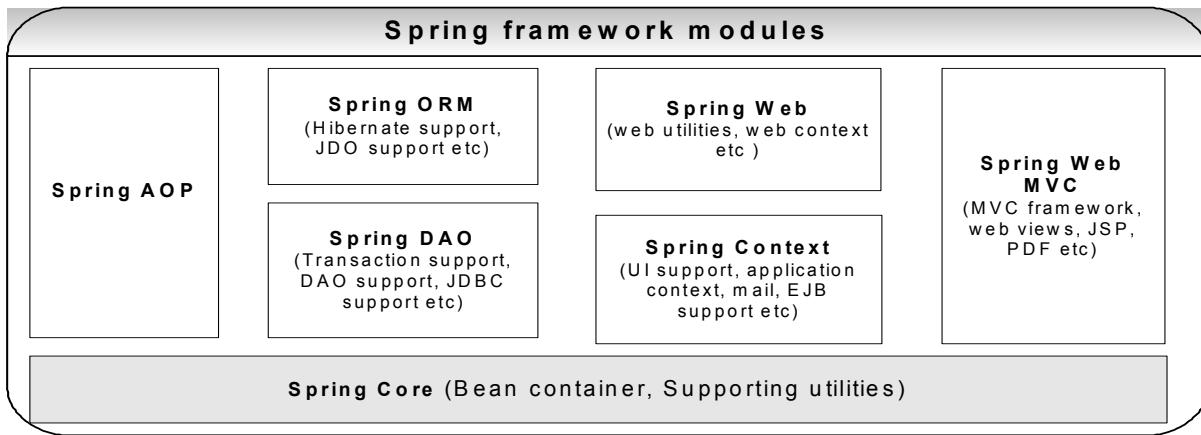
```
SELECT <fields from Employee & Department> FROM employee
left outer join department on employee.departmentId = department.id where 1=1;
```

**Q 17:** Give an overview of the Spring framework?

**A 17:** The Spring framework is an open source and comprehensive framework for enterprise Java development. Unlike other frameworks, Spring does not impose itself on the design of a project due to its modular nature and, it has been divided logically into independent packages, which can function independently.

It includes abstraction layers for transactions, persistence frameworks, Web development, a JDBC integration framework, an AOP integration framework, email support, web services support etc. It also provides integration modules for popular Object-to-Relational (O/R) mapping tools like Hibernate, JDO etc. The designers of an application can feel free to use just a few Spring packages and leave out the rest. The other spring packages can be introduced into an existing application in a phased manner. Spring is based on the IOC pattern (aka Dependency Injection pattern) and also complements OOP (Object Oriented Programming) with AOP (Aspect Oriented Programming). You do not have to use AOP if you do not want to and AOP complements Spring IoC to provide a better middleware solution.

As shown in the diagram above the Spring modules are built on top of the core container, which defines how beans are configured, created and managed.



**Core Container** provides the essential basic functionality. The basic package in the spring framework is the org.springframework.beans package. The Spring framework uses JavaBeans and there are two ways in which clients can use the functionality of Spring framework -- BeanFactory and ApplicationContext. BeanFactory applies the **IOC pattern** and separates an application's configuration and dependency specification from the actual application code. **Collection of beans, holds bean definitions, auto-instantiates beans at runtime**

**Spring Application Context** is a configuration file that provides context information to Spring framework. The ApplicationContext builds on top of the BeanFactory and inherits all the basic features of Spring framework. In addition to basic features, ApplicationContext provides additional features like event management, internalization support, resource management, JNDI, EJB, email, and scheduling functionality. BeanFactory is useful in low memory situations, which does not have the excess baggage an ApplicationContext has. ApplicationContext assist the user to use Spring in a framework oriented way while the BeanFactory offers a programmatic approach.

**Spring AOP** enhances the Spring middleware support by providing declarative services. This allows any object managed by Spring framework to be AOP enabled. Spring AOP provides "declarative transaction management service" similar to transaction services provided by EJB. So with Spring AOP you can incorporate declarative transaction management without having to rely on EJB. AOP functionality is also fully integrated into Spring for logging and various other features.

**Spring DAO** uses org.springframework.jdbc package to provide all the JDBC related support required by your application. This abstraction layer offers a meaningful hierarchy for handling exceptions and errors thrown by different database vendors with the help of Spring's *SQLExceptionTranslator*. So this abstraction layer simplifies error handling and greatly reduces the amount of exception handling code you need to write. It also handles opening and closing of connections.

**Spring ORM** framework is built on top of Spring DAO and it plugs into several object-to-relational (ORM) mapping tools like Hibernate, JDO, etc. Spring provides very good support for Hibernate by supporting Hibernate sessions, Hibernate transaction management etc.

**Spring Web** sits on top of the ApplicationContext module to provide context for Web based applications. This provides integration with Struts MVC framework. Spring Web module also assists with binding HTTP request parameters to domain objects and eases the tasks of handling multipart requests.

**Spring MVC** framework provides a pluggable MVC architecture. The users have a choice to use this framework or continue to use other frameworks like Struts. Spring separates the roles of controller, model object, dispatcher and handler object, which makes it easier to customize. Spring Web framework does not force user to use only JSP, but accommodates various view technologies like XSLT, velocity templates, Tiles, etc.

Spring framework is a modular framework, which uses complementary technologies such as IoC and AOP to be used in complex enterprise application development. Spring functionality can be used in any J2EE server.

**Q 18:** How would EJB 3.0 simplify your Java development compared to EJB 1.x, 2.x?

**A 18:** EJB 3.0 is taking ease of development very seriously and has adjusted its model to offer the POJO (Plain Old Java Object) persistence and the new **O/R mapping model inspired by and based on Hibernate** (a less intrusive model). In EJB 3.0, **all kinds of enterprise beans are just POJOs**. EJB 3.0 **extensively uses Java annotations**, which replace excessive XML based configuration files and eliminate the need for rigid component

model used in EJB 1.x, 2.x. Annotations can be used to define a bean's business interface, O/R mapping information, resource references etc.

- In EJB 1.x, 2.x the container manages the behaviour and internal state of the bean instances at runtime. All the EJB 1.x, 2.x beans must adhere to a rigid specification. In EJB 3.0, all container services can be configured and delivered to any POJO in the application via annotations. **You can build complex object structures with POJOs. Java objects can inherit from each other.** EJB 3.0 components are only coupled via their published business interfaces hence the implementation classes can be changed without affecting rest of the application. This makes the application more robust, easier to test, more portable and makes it easier to build loosely coupled business components in POJO.
- EJB 3.0 unlike EJB 1.x, 2.x **does not have a home interface**. The bean class may or may not implement a business interface. If the bean class does not implement any business interface, a business interface will be generated using the public methods. If only certain methods should be exposed in the business interface, all of those methods can be marked with @BusinessMethod annotation.
- EJB 3.0 defines smart default values. For example by default all generated interfaces are local, but the @Remote annotation can be used to indicate that a remote interface should be generated.
- EJB 3.0 supports both unidirectional and bidirectional relationships between entities.
- EJB 3.0 makes use of dependency injection to make decoupled service objects and resources like queue factories, queues etc available to any POJO. Using the @EJB annotation, you can inject an EJB stub into any POJO managed by the EJB 3.0 container and using @Resource annotation you can inject any resource from the JNDI.
- EJB 3.0 wires runtime services such as transaction management, security, logging, profiling etc to applications at runtime. Since those services are not directly related to application's business logic they are not managed by the application itself. Instead, the services are transparently applied by the container utilizing AOP (Aspect Oriented Programming). To apply a transaction attribute to a POJO method using annotation:

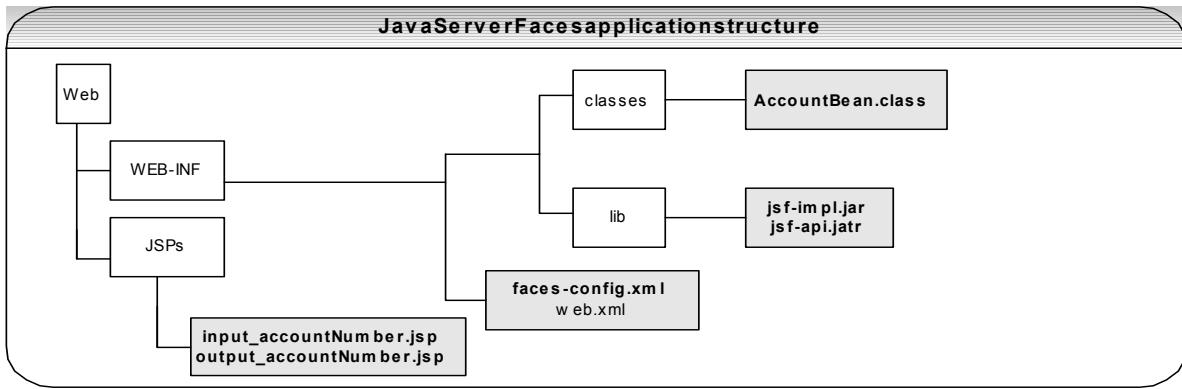
```
public class Account {
 @TransactionAttribute(TransactionAttributeType.REQUIRED)
 public getAccountDetails(){
 ...
 }
}
```

- EJB QL queries can be defined through the @NamedQuery annotation. You can also create regular JDBC style queries using the *EntityManager*. POJOs are not persistent by birth and become persistent once it is associated with an *EntityManager*.

**Q 19:** Briefly explain key features of the JavaServer Faces (JSF) framework?

**A 19:** JavaServer Faces is a new framework for building Web applications using java. JSF provides you with the following main features:

- Basic user interface components like buttons, input fields, links etc. and custom components like tree/table viewer, query builder etc. JSF was built with a component model in mind to allow tool developers to support Rapid Application Development (RAD). User interfaces can be created from these reusable server-side components.
- Provides a set of JSP tags to access interface components. Also provides a framework for implementing custom components.
- Supports mark up languages other than HTML like WML (Wireless Markup Language) etc by encapsulating event handling and component rendering. There is a single controller servlet every request goes through where the job of the controller servlet is to receive a faces page with components and then fire off events for each component render the components using a render tool kit.
- Uses a declarative navigation model by defining the navigation rules inside the XML configuration file faces-config.xml. This configuration file also defines bean resources used by JSF.
- JSF can hook into your model, which means the model is loosely coupled from JSF.



Let's look at some code snippets. Texts are stored in a properties file called **message.properties** so that this properties file can be quickly modified without having to modify the JSPs and also more maintainable because multiple JSP pages can use the same property.

```
account_nuber = Account number
account_button = Get account details
account_message=Processing account number :
```

#### **input\_accountNumber.jsp**

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<f:loadBundle basename="messages" var="msg"/>

<html>
...
<body>
<f:view>
<h:form id="accountForm">
<h:outputText value="#{msg.account_number}" />
<h:inputText value="#{accountBean.accountNumber}" />
<h:commandButton action="getAccount" value="#{msg.account_button}" />
</h:form>
</f:view>
</body>
</html>
```

#### **AccountBean.Java**

```
public class AccountBean {
 String accountNumber;

 public String getAccountNumber() {
 return accountNumber;
 }

 public void setAccountNumber(String accountNumber) {
 this.accountNumber = accountNumber;
 }
}
```

#### **faces-config.xml**

```
...
<faces-config>

<navigation-rule>
<form-view-id>/jsp/input_accountNumber.jsp</form-view-id>
<navigation-case>
<from-outcome>getAccount</from-outcome>
<to-view-id>/jsp/output_accountNumber.jsp</to-view-id>
</navigation-case>
</navigation-rule>
```

```

...
<managed-bean>
 <managed-bean-name>accountBean</managed-bean-name>
 <managed-bean-class>AccountBean</managed-bean-class>
 <managed-bean-scope>request</managed-bean-scope>
</managed-bean>

</faces-config>

```

**output\_accountNumber.jsp**

```

<html>
...
<body>
 <f:view>
 <h3>
 <h:outputText value="#{msg.account_message}" />
 <h:outputText value="#{accountBean.accountNumber}" />
 </h3>
 </f:view>
</body>
</html>

```

**Q 20:** How would the JSF framework compare with the Struts framework?

**A 20:**

Struts framework	JavaServer Faces
Matured since Struts has been around for a few years. It has got several successful implementations.	JSF is in its early access release and as a result somewhat immature.
The heart of Struts framework is the Controller, which uses the Front Controller design pattern and the Command design pattern. Struts framework has got <b>only single event handler</b> for the HTTP request.	The heart of JSF framework is the Page Controller Pattern where there is a front controller servlet where all the faces request go through with the UI components and then fire off events for each component and render the components using a render toolkit. So JSF can have <b>several event handlers on a page</b> . Also JSF loosely couples your model, where it can hook into your model (i.e unlike Struts your model does not have to extend JSF classes).
Struts does not have the vision of Rapid Application Development (RAD).	JSF was built with a component model in mind to allow RAD. JSF can be thought of as a combination of Struts framework for thin clients and the Java Swing user interface framework for thick clients.
Has got flexible page navigation using navigation rules inside the struts-config.xml file and Action classes using maoping.findForward(...).	JSF allows for more flexible navigation and a better design because the navigation rule (specified in <b>faces-config.xml</b> ) is decoupled from the Action whereas Struts forces you to hook navigation into your Action classes.
Struts is a sturdy frame work which is extensible and flexible. The existing Struts applications can be migrated to use JSF component tags instead of the original Struts HTML tags because Struts tags are superseded and also not undergoing any active development. You can also use the <b>Struts-Faces Integration</b> library to migrate your pages one page at a time to using JSF component tags.	JSF is more flexible than Struts because it was able to learn from Struts and also extensible and can integrate with RAD tools etc. So JSF will be a good choice for new applications.

So far we have discussed some of the emerging paradigms (IOC (aka dependency injection), AOP, Annotations Oriented Programming, and O/R mapping) and some of the frameworks, which are based on these paradigms. These paradigms and frameworks simplify your programming model by hiding the complexities behind the framework and minimising the amount of code an application developer has to write. JSF is also gathering lot of momentum and popularity as a Web tier UI framework in new J2EE applications.

## **SECTION FIVE**

### **Sample interview questions...**

#### **Tips:**

- Try to find out the needs of the project in which you will be working and the needs of the people within the project.
- 80% of the interview questions are based on your own resume.
- Where possible briefly demonstrate how you applied your skills/knowledge in the key areas as described in this book. Find the right time to raise questions and answer those questions to show your strength.
- Be honest to answer technical questions, you are not expected to know everything (for example you might know a few design patterns but not all of them etc).
- Do not be critical, focus on what you can do. Also try to be humorous.
- Do not act in superior way.

**Java**

Questions	Hint
<b>Multi-threading</b>	
What language features are available to allow shared access to data in a multi-threading environment?	Synchronized block, Synchronized method, wait, notify
What is the difference between synchronized method and synchronized block? When would you use?	Block on subset of data. Smaller code segment.
What Java language features would you use to implement a producer (one thread) and a consumer (another thread) passing data via a stack?	wait, notify
<b>Data Types</b>	
What Java classes are provided for date manipulation?	Calendar, Date
What is the difference between String and StringBuffer?	mutable, efficient
How do you ensure a class is Serializable?	Implement Serializable
What is the difference between static and instance field of a class	Per class vs. Per Object
What method do you need to implement to store class in Hashtable or HashMap?	hashCode(), equals()
How do you exclude a field of the class from serialization?	transient
<b>Inheritance</b>	
What is the difference between an Interface and an abstract base class?	interface inheritance, implementation inheritance.
What does overriding a method mean? (What about overloading?)	inheritance (different signature)
<b>Memory</b>	
What is the Java heap, and what is the stack?	dynamic, program thread execution.
Why does garbage collection occur and when can it occur?	To recover memory, as heap gets full.
If I have a circular reference of objects, but I no longer reference any of them from any executing thread, will these cause garbage collection problems?	no
<b>Exceptions</b>	
What is the problem or benefits of catching or throwing type "java.lang.Exception"?	Hides all subsequent exceptions.
What is the difference between a runtime exception and a checked exception?	Must catch or throw checked exceptions.

**Web components**

Questions	HINT
<b>JSP</b>	
What is the best practice regarding the use of scriptlets in JSP pages? (Why?)	Avoid
How can you avoid scriptlet code?	custom tags, Java beans
What do you understand by the term JSP compilation?	compiles to servlet code
<b>Servlets</b>	
What does Servlet API provide to store user data between requests?	HttpSession
What is the difference between forwarding a request and redirecting?	redirect return to browser
What object do you use to forward a request?	RequestDispatcher
What do you need to be concerned about with storing data in a servlet instance fields?	Multi-threaded.
What's the requirement on data stored in HttpSession in a clustered (distributable) environment?	Serializable
If I store an object in session, then change its state, is the state replicated to distributed Session?	No, only on setAttribute() call.
How does URL-pattern for servlet work in the web.xml?	/ddd/* or *.jsp
What is a filter, and how does it work?	Before/after request, chain.

**Enterprise**

Questions	Hint
<b>JDBC</b>	
What form of statement would you use to include user-supplied values?	PreparedStatement
Why might a PreparedStatement be more efficient than a statement?	Execution plan cache.
How would you prevent an SQL injection attack in JDBC?	PreparedStatement
What is the performance impact of testing against NULL in WHERE clause on Oracle?	Full table scan.
List advantages and disadvantages in using stored procedures?	Pro: integration with existing dbase, reduced network traffic Con: not portable, multiple language knowledge required
What is the difference between sql.Date, sql.Time, and sql.Timestamp?	Date only, time only, date and time

If you had a missing int value how do you indicate this to PreparedStatement?	setNull(pos, TYPE)
How can I perform multiple inserts in one database interaction?	executeBatch
Given this problem: Program reads 100,000 rows, converts to Java class in list, then converts list to XML file using reflection. Runs out of program memory. How would you fix?	Read one row at time, limit select, allocate more heap (result set = cursor)
How might you model object inheritance in database tables?	Table per hierarchy, table per class, table per concrete class
<b>JNDI</b>	
What are typical uses for the JNDI API within an enterprise application	Resource management, LDAP access
Explain the difference between a lookup of these "java:comp/env/ejb/MyBean" and "ejb/MyBean"?	logical mapping performed for java:comp/env
What is difference between new InitialContext() from servlet or from an EJB?	Different JNDI environments initialised. ejb controller by ejb-jar.xml, servlet by web.xml
What is an LDAP server used for in an enterprise environment?	authentication, authorisation
What is authentication, and authorisation?	Confirming identity, confirming access rights
<b>EJB</b>	
What is the difference between Stateless and Stateful session beans (used?)	Stateless holds per client state
What is the difference between Session bean and Entity bean (when used?)	Entity used for persistence
With Stateless Session bean pooling, when would a container typically take a instance from the pool and when would it return it?	for each business method
What is difference between "Required", "Supports", "RequiresNew" "NotSupported", "Mandatory", "Never"	Needs transaction, existing OK but doesn't need, must start new one, suspends transaction, must already be started, error if transaction
What is "pass-by-reference" and "pass-by-value", and how does it affect J2EE applications?	Reference to actual object versus copy of object. RMI pass by value
What EJB patterns, best practices are you aware of? Describe at least two.	Façade, delegate, value list, DAO, value object
How do you define finder methods for a CMP?	Home, xml
If I reference an EJB from another EJB what can I cache to improve performance, and where should I do the caching?	Home, set it up in setSessionContext
Describe some issues/concerns you have with the J2EE specification	Get their general opinion of J2EE
Why is creating field value in setSessionContext of a performance benefit?	pooled, gc
What is difference between System exception and application exception from an EJB method?	System exception, container will auto rollback
What do you understand by the term "offline optimistic locking" or long-lived business transaction? How might you implement this using EJB?	version number, date, field comparisons
Explain performance difference between getting a list of summary information (e.g. customer list) via finder using a BMP entity vs Session using DAO?	BMP: n+1 database reads, n rmi calls
What is meant by a coarse-grained and a fine-grained interface?	Amount of data transferred per method call
<b>XML/XSLT</b>	
What is the difference between a DOM parser and a SAX parser?	DOM: reads entire model, SAX: event published during parsing
What is difference between DTD and XML Schema?	level of detail, Schema is in xml.
What does the JAXP API do for you?	Parser independence
What is XSLT and how can it be used?	xml translation
What would be the XPath to select any element called table with the class attribute of info?	Table[@class='info']
<b>JMS</b>	
How can asynchronous events be managed in J2EE?	JMS
How do transactions affect the onMessage() handling of a MDB?	Taking off queue
If you send a JMS message from an EJB, and transaction rollback, will message be sent?	yes
How do you indicate what topic or queue MDB should react to?	deployment descriptor
What is difference between a topic and a queue?	broadcast, single
<b>SOAP</b>	
What is a Web service, and how does it relate to SOAP?	SOAP is the protocol
What is a common transport for SOAP messages?	HTTP
What is WSDL? How would you use a WSDL file?	XML description of Web Service: interface and how to bind to it.
With new J2EE SOAP support what is: JAXR, JAX-RPC, and SAAJ?	registry, rpc, attachments
<b>Security</b>	
Where can container level security be applied in J2EE application?	Web URI's, ejb methods
How can the current user be obtained in a J2EE application (Web and Enterprise)?	getUserPrincipal getCallerPrincipal
How can you perform role checks in a J2EE application (Web and enterprise)?	IsUserInRole()

	IsCallerInRole()
<b>Design</b>	
Questions	Hint
<b>OO</b>	
Name some type of UML diagrams.	class, sequence, activity, use case
Describe some types of relationships can you show on class diagrams?	generalisation, aggregation, uses
What is difference between association, aggregation, and generalisation?	Relationship, ownership, inheritance
What is a sequence diagram used to display?	Object instance interactions via operations/signals
What design patterns do you use. Describe one you have used (not singleton)	e.g. Builder, Factory, Visitor, Chain of Command
Describe the observer pattern and an example of how it would be used	e.g. event notification when model changes to view
What are Use Cases?	Define interaction between actors and the system
What is your understanding of encapsulation?	Encapsulate data and behaviour within class
What is your understanding of polymorphism?	Class hierarchy, runtime determine instance
<b>Process</b>	
Have you heard of or used test-driven development?	e.g. XP process
What previous development process have you followed?	Rational, XP, waterfall
How do you approach capturing client requirements?	Numbered requirements, use case
What process steps would you include between the capture of requirements and when coding begins?	Architecture, Design, UML modelling
How would you go about solving performance issue in an application?	Set goals, establish bench, profile application, make changes one at a time
What developer based testing are you familiar with (before system testing?)	Unit test discussion
How might you test a business system exposed via a Web interface?	Automated script emulating browser
What is your experience with iterative development?	Multiple iteration before release
<b>Distributed Application</b>	
Explain a typical architecture of a business system exposed via Web interface?	Explain tiers (presentation, enterprise, resource) Java technology used in each tiers, hardware distribution of Web servers, application server, database server
Describe what tiers you might use in a typical large scale (> 200 concurrent users) application and the responsibilities of each tier (where validation, presentation, business logic, persistence occur).	Another way of asking same question as above if their answer wasn't specific enough
Describe what you understand by being able to "scale" an application? How does a J2EE environment aid scaling.	Vertical and Horizontal scaling. Thread management, clustering, split tiers
What are some security issues in Internet based applications?	authentication, authorisation, data encryption, denial service, xss attacks
<b>General</b>	
Questions	Hints
What configuration management are you familiar with?	e.g. CVS, ClearCase
What issue/tracking process have you followed?	Want details on bug recording and resolution process.
What are some key factors to working well within a team?	Gets a view on how you would work within interviewers' environment.
What attributes do you assess when considering a new job? (what makes it a good one)	Insight into what motivates you.
What was the last computing magazine you read? Last computing book? What is a regular online magazine/reference you use?	Understand how up to date you keep yourself.

**GLOSSARY OF TERMS**

<b>TERM</b>	<b>DESCRIPTION</b>
ACID	Atomicity, Consistency, Isolation, Duration.
aka	Also known as.
AOP	Aspect Oriented Programming
API	Application Program Interface
AWT	Abstract Window Toolkit
BLOB	Binary Large Object
BMP	Bean Managed Persistence
CGI	Common Gateway Interface
CLOB	Character Large OBject
CMP	Container Managed Persistence
CORBA	Common Object Request Broker Architecture
CRM	Customer Relationships Management
CSS	Cascading Style Sheets
DAO	Data Access Object
DNS	Domain Name Service
DOM	Document Object Model
DTD	Document Type Definition
EAR	Enterprise ARchive
EIS	Enterprise Information System
EJB	Enterprise JavaBean
ERP	Enterprise Resource Planning
FDD	Feature Driven Development
GIF	Graphic Interchange Format
GOF	Gang Of Four
HQL	Hibernate Query Language.
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
I/O	Input/Output
IDE	Integrated Development Environment
IIOP	Internet Inter-ORB Protocol
IoC	Inversion of Control
IP	Internet Protocol
J2EE	Java 2 Enterprise Edition
JAAS	Java Authentication and Authorization Service
JAF	JavaBeans Activation Framework
JAR	Java ARchive
JAXB	Java API for XML Binding
JAXP	Java API for XML Parsing
JAXR	Java API for XML Registries
JAX-RPC	Java API for XML-based RPC
JAX-WS	Java API for XML-based Web Services
JCA	J2EE Connector Architecture
JDBC	Java Database Connectivity
JDK	Java Development Kit
JMS	Java Messaging Service
JMX	Java Management eXtensions
JNDI	Java Naming and Directory Interface
JNI	Java Native Interface
JRMP	Java Remote Method Protocol
JSF	JavaServer Faces
JSP	Java Server Pages
JSTL	Java Standard Tag Library
JTA	Java Transaction API
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
MOM	Message Oriented Middleware
MVC	Model View Controller
NDS	Novell Directory Service
NIO	New I/O
O/R mapping	Object to Relational mapping.
OO	Object Oriented

OOP	Object Oriented Programming
OOPL	Object Oriented Programming Language
ORB	Object Request Broker
ORM	Object to Relational Mapping.
POJI	Plain Old Java Interface
POJO	Plain Old Java Object
RAR	Resource adapter ARchive
RDBMS	Relational Database Management System
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RUP	Rational Unified Process
SAAJ	SOAP with attachment API for Java
SAX	Simple API for XML
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TDD	Test Driven Development
UDDI	Universal Description Discovery and Integration
UDP	User Datagram Protocol
UI	User Interface
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTF	
VO	Value Object which is a plain Java class which has attributes or fields and corresponding getter → getXXX() and setter → setXXX() methods .
WAR	Web ARchive
WSDL	Web Service Description Language
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XP	Extreme Programming
XPath	XML Path
XSD	XML Schema Definition
XSL	Extensible Style Language
XSL-FO	Extensible Style Language – Formatting Objects
XSLT	Extensible Style Language Transformation

---

## RESOURCES

---

### Articles

- Sun Java Certified Enterprise Architect by Leo Crawford on <http://www.leocrawford.org.uk/work/jcea/part1/index.html>.
- Practical UML: A Hands-On Introduction for Developers by Randy Miller on <http://bdn.borland.com/article/0,1410,31863,00.html>
- W3 Schools on <http://www.w3schools.com/default.asp>.
- LDAP basics on <http://publib.boulder.ibm.com/iseries/v5r2/ic2924/index.htm?info/rzahy/rzahyovrc0.htm>.
- Java World articles on design patterns: <http://www.javaworld.com/columns/jw-Java-design-patterns-index.shtml>.
- Web Servers vs. App Servers: Choosing Between the Two By Nelson King on <http://www.serverwatch.com/tutorials/article.php/1355131>.
- Follow the Chain of Responsibility by David Geary on Java World - <http://www.javaworld.com/javaworld/jw-08-2003/jw-0829-designpatterns.html>.
- J2EE Design Patterns by Sue Spielman on <http://www.onjava.com/pub/a/onjava/2002/01/16/patterns.html>.
- The New Methodology by Martin Fowler on <http://www.martinfowler.com/articles/newMethodology.html>.
- Merlin brings nonblocking I/O to the Java platform by Aruna Kalagnanam and Balu G on <http://www.ibm.com/developerworks/Java/library/j-javaio>.
- Hibernate Tips and Pitfalls by Phil Zoio on <http://www.realsolve.co.uk/site/tech/hib-tip-pitfall-series.php>.
- Hibernate Reference Documentation on [http://www.hibernate.org/hib\\_docs/reference/en/html\\_single/](http://www.hibernate.org/hib_docs/reference/en/html_single/).
- Object-relation mapping without the container by Richard Hightower on <http://www-128.ibm.com/developerworks/library/j-hibern/?ca=dnt515>.
- Object to Relational Mapping and Relationships with Hibernate by Mark Eagle on <http://www.meagle.com:8080/hibernate.jsp>.
- Mapping Objects to Relational databases: O/R Mapping In detail by Scott W. Ambler on <http://www.agiledata.org/essays/mappingObjects.html>.
- I want my AOP by Ramnivas Laddad on Java World.
- Websphere Application Server 5.0 for iSeries – Performance Considerations by Jill Peterson.
- Dependency Injection using pico container by Subbu Ramanathan .
- Websphere Application Server & Database Performance tuning by Michael S. Pallos on <http://www.bizforum.org/whitepapers/candle-5.htm>.
- A beginners guide to Dependency Injection by Dhananjay Nene on <http://www.theserverside.com/articles/article.tss?I=IOCBeginners>.
- The Spring series: Introduction to the Spring framework by Naveen Balani on <http://www-128.ibm.com/developerworks/web/library/wa-spring1>.
- The Spring Framework by Benoy Jose.
- Inversion of Control Containersband the Dependency Injection pattern by Martin Fowler.
- Migrate J2EE Applications for EJB 3.0 by Debu Panda on JavaPro.
- EJB 3.0 in a nutshell by Anil Sharma on JavaWorld.
- Preparing for EJB 3.0 by Mike Keith on ORACLE Technology Network.
- Simplify enterprise Java development with EJB 3.0 by Michael Juntao Yuan on JavaWorld.
- J2SE: New I/O by John Zukowski on <http://java.sun.com/developer/technicalArticles/releases/nio/>.

- High-Performance I/O arrives by Danniell F. Savarese on JavaPro.
- Hibernate – Proxy Visitor Pattern by Kurtis Williams.
- Best Practices for Exception Handling by Gunjan Doshi.
- Three Rules for Effective Exception Handling by Jim Cushing.
- LDAP and JNDI: Together forever – by Sameer Tyagi.
- Introduction To LDAP – by Brad Marshall.
- Java theory and practice: Decorating with dynamic proxies by Brian Goetz.
- Java Dynamic Proxies: One Step from Aspect-Oriented Programming by Lara D'Abreo.
- Java Design Patterns on [http://www.allapplabs.com/java\\_design\\_patterns](http://www.allapplabs.com/java_design_patterns) .
- Software Design Patterns on <http://www.dofactory.com/Patterns/Patterns.aspx> .
- JRun: Core Dump and Dr. Watson Errors on [http://www.macromedia.com/cfusion/knowledgebase/index.cfm?id=tn\\_17534](http://www.macromedia.com/cfusion/knowledgebase/index.cfm?id=tn_17534)
- [www.javaworld.com](http://www.javaworld.com) articles.
- <http://www-128.ibm.com/developerworks/java> articles.
- <http://www.devx.com/java> articles.
- [www.theserverside.com/tss](http://www.theserverside.com/tss) articles.
- <http://javaboutique.internet.com/articles> articles.

#### Books

- Beginning Java 2 by Ivor Horton.
- Design Patterns by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (GoF) .
- UML Distilled by Martin Fowler, Kendall Scott .
- Mastering Enterprise Java Beans II by Ed Roman, Scott Ambler, Tyler Jewell, Floyd Marinescu [download for free] .
- EJB Design Patterns by Floyd Marinescu [download for free] .
- Sun Certified Enterprise Architect for J2EE Technology Study Guide by Mark Cade and Simon Roberts.
- Professional Java Server Programming - J2EE edition by Wrox publication.
- Design Patterns Java Companion by James W. Cooper (Free download: <http://www.patterndepot.com/put/8/JavaPatterns.htm>).
- Test Driven Development – By Example, by Kent Beck.

---

**INDEX**

---

**Emerging Technologies/Frameworks**

Briefly explain key features of the JavaServer Faces (JSF) framework?	223
Explain Object-to-Relational (O/R) mapping?	218
Explain some of the pitfalls of Hibernate and explain how to avoid them?	220
Give an overview of hibernate framework?	218
Give an overview of the Spring framework?	221
How would EJB 3.0 simplify your Java development compared to EJB 1.x, 2.x?	222
How would the JSF framework compare with the Struts framework?	225
What are the benefits of IOC (aka Dependency Injection)?	217
What are the differences between OOP and AOP?	214
What are the different types of dependency injections?	217
What are the pros and cons of annotations over XML based deployment descriptors?	215
What is aspect oriented programming? Explain AOP?	212
What is attribute or annotation oriented programming?	215
What is inversion of control (IOC) (also known as dependency injection)?	216
What is Test Driven Development (TDD)?	211
What is the difference between a service locator pattern and an inversion of control pattern?	217
What is the point of Test Driven Development (TDD)?	211
What is XDoclet?	216
Why dependency injection is more elegant than a JNDI lookup to decouple client and the service?	218

**Enterprise - Best practices and performance considerations**

Explain some of the J2EE best practices to improve performance?	141
Explain some of the J2EE best practices?	139
Give some tips on J2EE application server performance tuning?	139

**Enterprise - EJB 2.x**

Can an EJB client invoke a method on a bean directly?	99
Discuss EJB container security?	105
Explain EJB architecture?	96
Explain exception handling in EJB?	103
Explain lazy loading and dirty marker strategies?	109
How can we determine if the data is stale (for example when using optimistic locking)?	104
How do you rollback a container managed transaction in EJB?	103
How to design transactional conversations with session beans?	102
What are EJB best practices?	106
What are isolation levels?	101
What are not allowed within the EJB container?	105
What are the implicit services provide by an EJB container?	101
What are transactional attributes?	101
What is a business delegate? Why should you use a business delegate?	107
What is a distributed transaction? What is a 2-phase commit?	102
What is a fast-lane reader?	109
What is a Service Locator?	109
What is a session façade?	108
What is a value object pattern?	108
What is dooming a transaction?	102

What is the difference between Container Managed Persistence (CMP) and Bean Managed Persistence (BMP)?	99
What is the difference between EJB 1.1 and EJB 2.0? What is the difference between EJB 2.x and EJB 3.0?	100
What is the difference between EJB and JavaBeans?	95
What is the difference between optimistic and pessimistic concurrency control?	104
What is the difference between session and entity beans?	99
What is the difference between stateful and stateless session beans?	99
What is the role of EJB 2.x in J2EE?	95

**Enterprise - J2EE**

Explain J2EE class loaders?	68
Explain MVC architecture relating to J2EE?	63
Explain the J2EE 3-tier or n-tier architecture?	61
So what is the difference between a component and a service you may ask?	60
What are ear, war and jar files? What are J2EE Deployment Descriptors?	64
What is J2EE? What are J2EE components and services?	60
What is the difference between a Web server and an application server?	64
Why use design patterns in a J2EE application?	64

**Enterprise - JDBC**

Explain differences among java.util.Date, java.sql.Date, java.sql.Time, and java.sql.Timestamp?	86
How to avoid the "running out of cursors" problem?	85
What are JDBC Statements? What are different types of statements? How can you create them?	83
What is a Transaction? What does setAutoCommit do?	84
What is JDBC? How do you connect to a database?	83
What is the difference between JDBC-1.0 and JDBC-2.0? What are Scrollable ResultSets, Updateable ResultSets, RowSets, and Batch updates?	85

What is the difference between statements and prepared statements?	86
--------------------------------------------------------------------	----

**Enterprise - JMS**

Discuss some of the design decisions you need to make regarding your message delivery?	112
Give an example of a J2EE application using Message Driven Bean with JMS?	114
How JMS is different from RPC?	110
What are some of the key message characteristics defined in a message header?	111
What is Message Oriented Middleware? What is JMS?	110
What type of messaging is provided by JMS?	111

**Enterprise - JNDI & LDAP**

Explain the difference between the look up of "java comp/env/ejb/MyBean" and "ejb/MyBean"?	88
Explain the RMI architecture?	90
How will you pass parameters in RMI?	93
What are the differences between RMI and a socket?	92
What are the services provided by the RMI Object?	92
What is a JNDI InitialContext?	88
What is a remote object? Why should we extend UnicastRemoteObject?	91
What is an LDAP server? And what is it used for in an enterprise environment?	88
What is HTTP tunnelling or how do you make RMI calls across firewalls?	93

What is JNDI? And what are the typical uses within a J2EE application?	87	How do you make a Servlet thread safe? What do you need to be concerned about with storing data in Servlet instance fields?	72
What is the difference between RMI and CORBA?	92	HTTP is a stateless protocol, so how do you maintain state? How do you store user data between requests?	69
Why use LDAP when you can do the same with relational database (RDBMS)?	89		
<b>Enterprise - JSP</b>		If an object is stored in a session and subsequently you change the state of the object, will this state change replicated to all the other distributed sessions in the cluster?	74
Explain hidden and output comments?	80	What are the considerations for servlet clustering?	73
Explain the life cycle methods of a JSP?	78	What are the ServletContext and ServletConfig objects? What are Servlet environment objects?	72
How will you avoid scriptlet code in JSP?	83	What is a filter, and how does it work?	74
Is JSP variable declaration thread safe?	80	What is a RequestDispatcher? What object do you use to forward a request?	73
Tell me about JSP best practices?	82	What is pre-initialization of a Servlet?	73
What are custom tags? Explain how to build custom tags?	81	What is the difference between CGI and Servlet?	69
What are implicit objects and list them?	79	What is the difference between doGet () and doPost () or GET and POST?	71
What are the differences between static and a dynamic include?	79	What is the difference between forwarding a request and redirecting a request?	73
What are the different scope values or what are the different scope values for <jsp:usebean> ?	79	What is the difference between HttpServlet and GenericServlet?	72
What are the main elements of JSP? What are scriptlets? What are expressions?	78	<b>Enterprise - Software development process</b>	
What is a JSP? What is it used for? What do you understand by the term JSP translation phase or compilation phase?	77	What software development processes/principles are you familiar with?	144
What is a TagExtraInfo class?	82	<b>Enterprise - SQL, Tuning and O/R mapping</b>	
What is the difference between custom JSP tags and Javabeans?	82	Explain a sub-query? How does a sub-query impact on performance?	121
<b>Enterprise - Logging, testing and deployment</b>		Explain inner and outer joins?	119
Enterprise - Logging, testing and deployment	143	How can you performance tune your database?	122
Give an overview of log4J?	141	How do you implement one-to-one, one-to-many and many-to-many relationships while designing tables?	122
How do you initialize and use Log4J?	142	How do you map inheritance class structure to relational data model?	123
What is the hidden cost of parameter construction when using Log4J?	142	How will you map objects to a relational database? How will you map class inheritance to relational data model?	122
What is the test phases and cycles?	143	What is a view? Why will you use a view? What is an aggregate function?	124
<b>Enterprise - Personal</b>		What is normalization? When to denormalize?	121
Have you used any load testing tools?	144	<b>Enterprise - Struts</b>	
Tell me about yourself or about some of the recent projects you have worked with? What do you consider your most significant achievement? Why do you think you are qualified for this position? Why should we hire you and what kind of contributions will you make?	144	Are Struts action classes thread-safe?	135
What operating systems are you comfortable with?	144	Give an overview of Struts?	133
What source control systems have you used?	144	How do you implement internationalization in Struts?	136
Which on-line technical resources do you use to resolve any design and/or development issues?	144	How do you upload a file in Struts?	135
<b>Enterprise - RUP &amp; UML</b>		What design patterns are used in Struts?	136
Explain the 4 phases of RUP?	127	What is a synchronizer token pattern in Struts or how will you protect your Web against multiple submissions?	135
What are the characteristics of RUP? Where can you use RUP?	128	What is an action mapping in Struts? How will you extend Struts?	136
What are the different types of UML diagrams?	128	<b>Enterprise - Web and Application servers</b>	
What is RUP?	126	Explain Java Management Extensions (JMX)?	138
What is the difference between a collaboration diagram and a sequence diagram?	133	What application servers, Web servers, LDAP servers, and Database servers have you used?	137
What is the difference between aggregation and composition?	133	What is a virtual host?	137
When to use 'use case' diagrams?	128	What is application server clustering?	138
When to use activity diagrams?	132	What is the difference between a Web server and an application server?	137
When to use class diagrams?	129	<b>Enterprise - Web and Applications servers</b>	
When to use interaction diagrams?	131	Explain some of the portability issues between different application servers?	139
When to use object diagrams?	130	<b>Enterprise - XML</b>	
When to use package diagrams?	130	What is the difference between a SAX parser and a DOM parser?	115
When to use statechart diagram?	131	What is XML? And why is XML important?	114
Why is UML important?	128	What is XPATH? What is XSLT/XSL/XSL-FO/XSD/DTD etc? What is JAXB? What is JAXP?	115
<b>Enterprise - Servlet</b>		Which is better to store data as elements or as attributes?	115
Briefly discuss the following patterns Composite view, View helper, Dispatcher view and Service to worker? Or explain J2EE design patterns?	76		
Explain declarative security for WEB applications?	74		
Explain Servlet URL mapping?	77		
Explain the directory structure of a WEB application?	71		
Explain the Front Controller design pattern or explain J2EE design patterns?	75		
Explain the life cycle methods of a servlet?	70		

**How would you go about...?**

- How would you go about applying the design patterns in your Java/J2EE application? 165  
 How would you go about applying the Object Oriented (OO) design concepts in your Java/J2EE application? 160  
 How would you go about applying the UML diagrams in your Java/J2EE project? 162  
 How would you go about describing the open source projects like JUnit (unit testing), Ant (build tool), CVS (version control system) and log4J (logging tool) which are integral part of most Java/J2EE projects? 199  
 How would you go about describing the software development processes you are familiar with? 163  
 How would you go about describing Web services? 206  
 How would you go about designing a Java/J2EE application? 153  
 How would you go about determining the enterprise security requirements for your Java/J2EE application? 194  
 How would you go about documenting your Java/J2EE application? 152  
 How would you go about identifying any potential thread-safety issues in your Java/J2EE application? 158  
 How would you go about identifying any potential transactional issues in your Java/J2EE application? 159  
 How would you go about identifying performance and/or memory issues in your Java/J2EE application? 156  
 How would you go about improving performance in your Java/J2EE application? 157  
 How would you go about minimising memory leaks in your Java/J2EE application? 157

**Java**

- Briefly explain high-level thread states? 38  
 Discuss the Java error handling mechanism? What is the difference between Runtime (unchecked) exceptions and checked exceptions? What is the implication of catching all the exceptions with the type "Exception"? 35  
 Explain different ways of creating a thread? 38  
 Explain Java class loaders? Explain dynamic class loading? 13  
 Explain Outer and Inner classes (or Nested classes) in Java? When will you use an Inner Class? 31  
 Explain static vs dynamic class loading? 13  
 Explain the assertion construct? 19  
 Explain the Java Collection framework? 21  
 Explain the Java I/O streaming concept and the use of the decorator design pattern in Java I/O? 26  
 Explain threads blocking on I/O? 41  
 Give a few reasons for using Java? 12  
 Give an example where you might use a static method? 29  
 How can threads communicate with each other? How would you implement a producer (one thread) and a consumer (another thread) passing data (via stack)? 40  
 How can you improve Java I/O performance? 28  
 How do you express an 'is a' relationship and a 'has a' relationship or explain inheritance and composition? What is the difference between composition and aggregation? 15  
 How does Java allocate stack and heap memory? Explain re-entrant, recursive and idempotent methods/functions? 31  
 How does the Object Oriented approach improve software development? 14  
 How does thread synchronization occurs inside a monitor? What levels of synchronization can you apply? What is the difference between synchronized method and synchronized block? 39  
 How will you call a Web server from a stand alone Java application? 44  
 If 2 different threads hit 2 different synchronized methods in an object at the same time will they both continue? 40  
 If you have a circular reference of objects, but you no longer reference it from an execution thread, will this

- object be a potential candidate for garbage collection? 34  
 What are "static initializers" or "static blocks with no function names"? 14  
 What are access modifiers? 30  
 What are some of the best practices relating to Java collection? 22  
 What are the advantages of Object Oriented Programming Languages (OOPL)? 14  
 What are the benefits of the Java collection framework? 22  
 What do you know about the Java garbage collector? When does the garbage collection occur? Explain different types of references in Java? 34  
 What do you mean by polymorphism, inheritance, encapsulation, and dynamic binding? 15  
 What is a daemon thread? 40  
 What is a factory pattern? 42  
 What is a final modifier? Explain other Java modifiers? 30  
 What is a singleton pattern? How do you code it in Java? 41  
 What is a socket? How do you facilitate inter process communication in Java? 43  
 What is a user defined exception? 37  
 What is design by contract? Explain the assertion construct? 18  
 What is serialization? How would you exclude a field of a class from serialization or what is a transient variable? 26  
 What is the common use? 26  
 What is the difference between aggregation and composition? 15  
 What is the difference between an abstract class and an interface and when should you use them? 20  
 What is the difference between an instance variable and a static variable? Give an example where you might use a static variable? 29  
 What is the difference between C++ and Java? 12  
 What is the difference between final, finally and finalize() in Java? 31  
 What is the difference between processes and threads? 37  
 What is the difference between yield and sleeping? 39  
 What is the main difference between a String and a StringBuffer class? 25  
 What is the main difference between an ArrayList and a Vector? What is the main difference between Hashmap and Hashtable? 21  
 What is the main difference between pass-by-reference and pass-by-value? 25  
 What is the main difference between shallow cloning and deep cloning of objects? 29  
 What is the main difference between the Java platform and the other software platforms? 12  
 What is type casting? Explain up casting vs down casting? When do you get ClassCastException? 33  
 When is a method said to be overloaded and when is a method said to be overridden? 21  
 When providing a user defined key class for storing objects in the Hashmaps or Hashtables, what methods do you have to provide or override (ie method overriding)? 24  
 When to use an abstract class? 20  
 When to use an interface? 21  
 Where and how can you use a private constructor? 30  
 Why is it not advisable to catch type "Exception"? 36  
 Why should you catch a checked exception late in a catch {} block? 36  
 Why should you throw an exception early? 36  
 Why there are some interfaces with no defined methods (i.e. marker interfaces) in Java? 21  
**Java - Applet**  
 How will you communicate between two Applets? 49  
 How will you initialize an applet? 48  
 How would you communicate between applets and servlets? 49  
 What is a signed Applet? 49

What is the difference between an applet and an application?	49	How do you handle pressure? Do you like or dislike these situations?	54
What is the order of method invocation in an applet?	48	Tell me about yourself or about some of the recent projects you have worked with? What do you consider your most significant achievement? Why do you think you are qualified for this position? Why should we hire you and what kind of contributions will you make?	53
<b>Java - Performance &amp; Memory leaks</b>		What are your career goals? Where do you see yourself in 5-10 years?	55
How would you detect and minimise memory leaks in Java?	51	What are your strengths and weaknesses? Can you describe a situation where you took initiative? Can you describe a situation where you applied your problem solving skills?	54
How would you improve performance of a Java application?	50	What do you like and/or dislike most about your current and/or last position?	54
Why does the JVM crash with a core dump or a Dr.Watson error?	52	What past accomplishments gave you satisfaction? What makes you want to work hard?	55
<b>Java - Swing</b>		What was the last Java related book or article you read?	55
Explain layout managers?	47	Why are you leaving your current position?	54
Explain the Swing Action architecture?	45	Why do you want to work for us?	55
Explain the Swing delegation event model?	48		
Explain the Swing event dispatcher mechanism?	46		
If you add a component to the CENTER of a border layout, which directions will the component stretch?	45		
What do you understand by MVC as used in a JTable?	47		
What is the base class for all Swing components?	45		
What is the difference between AWT and Swing?	44		
<b>Java/J2EE - Personal</b>			
Did you have to use any design patterns in your Java project?	53		
Do you have any role models in software development?	55		
		<b>Key Points</b>	
		Enterprise - Key Points	146
		Java - Key Points	56

# TOP 100 SOFTWARE TESTING

## 1. What is the MAIN benefit of designing tests early in the life cycle?

It helps prevent defects from being introduced into the code.

## 2. What is risk-based testing?

Risk-based testing is the term used for an approach to creating a test strategy that is based on prioritizing tests by risk. The basis of the approach is a detailed risk analysis and prioritizing of risks by risk level. Tests to address each risk are then specified, starting with the highest risk first.

## 3. A wholesaler sells printer cartridges. The minimum order quantity is 5. There is a 20% discount for orders of 100 or more printer cartridges. You have been asked to prepare test cases using various values for the number of printer cartridges ordered. Which of the following groups contain three test inputs that would be generated using Boundary Value Analysis?

4, 5, 99

## 4. What is the KEY difference between preventative and reactive approaches to testing?

Preventative tests are designed early; reactive tests are designed after the software has been produced.

## 5. What is the purpose of exit criteria?

The purpose of exit criteria is to define when a test level is completed.

## 6. What determines the level of risk?

The likelihood of an adverse event and the impact of the event determine the level of risk.

## 7. When is used Decision table testing?

Decision table testing is used for testing systems for which the specification takes the form of rules or cause-effect combinations. In a decision table the inputs are listed in a column, with the outputs in the same column but below the inputs. The remainder of the table explores combinations of inputs to define the outputs produced.

Learn More About Decision Table Testing Technique in the Video Tutorial [here](#)

## 8. What is the MAIN objective when reviewing a software deliverable?

To identify defects in any software work product.

**9. Which of the following defines the expected results of a test? Test case specification or test design specification.**

Test case specification defines the expected results of a test.

**10. What is the benefit of test independence?**

It avoids author bias in defining effective tests.

**11. As part of which test process do you determine the exit criteria?**

The exit criteria is determined on the bases of ‘Test Planning’ .

**12. What is beta testing?**

Testing performed by potential customers at their own locations.

**13. Given the following fragment of code, how many tests are required for 100% decision coverage?**

**if width > length**

**thenbiggest\_dimension = width**

**if height > width**

**thenbiggest\_dimension = height**

**end\_if**

**elsebiggest\_dimension = length**

**if height > length**

**thenbiggest\_dimension = height**

**end\_if**

**end\_if**

**14. You have designed test cases to provide 100% statement and 100% decision coverage for the following fragment of code. if width > length then biggest\_dimension = width else biggest\_dimension = length end\_if The following has been added to the bottom of the code fragment above. print "Biggest dimension is " &biggest\_dimensionprint "Width: " & width print "Length: " & length How many more test cases are required?**

None, existing test cases can be used.

## **15. Rapid Application Development?**

Rapid Application Development (RAD) is formally a parallel development of functions and subsequent integration. Components/functions are developed in parallel as if they were mini projects, the developments are time-boxed, delivered, and then assembled into a working prototype. This can very quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements. Rapid change and development of the product is possible using this methodology. However the product specification will need to be developed for the product at some point, and the project will need to be placed under more formal controls prior to going into production.

## **16. What is the difference between Testing Techniques and Testing Tools?**

Testing technique: – Is a process for ensuring that some aspects of the application system or unit functions properly there may be few techniques but many tools.

Testing Tools: – Is a vehicle for performing a test process. The tool is a resource to the tester, but itself is insufficient to conduct testing

Learn More About Testing Tools [here](#)

## **17. We use the output of the requirement analysis, the requirement specification as the input for writing ...**

User Acceptance Test Cases

## **18. Repeated Testing of an already tested program, after modification, to discover any defects introduced or uncovered as a result of the changes in the software being tested or in another related or unrelated software component:**

Regression Testing

## **19. What is component testing?**

Component testing, also known as unit, module and program testing, searches for defects in, and verifies the functioning of software (e.g. modules, programs, objects, classes, etc.) that are separately testable. Component testing may be done in isolation from the rest of the system depending on the context of the development life

cycle and the system. Most often stubs and drivers are used to replace the missing software and simulate the interface between the software components in a simple manner. A stub is called from the software component to be tested; a driver calls a component to be tested.

Here is an awesome video on [Unit Testing](#)

## 20. What is functional system testing?

Testing the end to end functionality of the system as a whole is defined as a functional system testing.

## 21. What are the benefits of Independent Testing?

Independent testers are unbiased and identify different defects at the same time.

## 22. In a REACTIVE approach to testing when would you expect the bulk of the test design work to be begun?

The bulk of the test design work begun after the software or system has been produced.

## 23. What are the different Methodologies in Agile Development Model?

There are currently seven different agile methodologies that I am aware of:

1. [Extreme Programming \(XP\)](#)
2. [Scrum](#)
3. [Lean Software Development](#)
4. [Feature-Driven Development](#)
5. Agile Unified Process
6. Crystal
7. Dynamic Systems Development Model (DSDM)

## 24. Which activity in the fundamental test process includes evaluation of the testability of the requirements and system?

A ‘Test Analysis’ and ‘Design’ includes evaluation of the testability of the requirements and system.

## 25. What is typically the MOST important reason to use risk to drive testing efforts?

Because testing everything is not feasible.

## 26. What is random/monkey testing? When it is used?

Random testing often known as monkey testing. In such type of testing data is generated randomly often using a tool or automated mechanism. With this randomly generated input the system is tested and results are analysed

accordingly. These testing are less reliable; hence it is normally used by the beginners and to see whether the system will hold up under adverse effects.

**27. Which of the following are valid objectives for incident reports?**

1. Provide developers and other parties with feedback about the problem to enable identification, isolation and correction as necessary.
2. Provide ideas for test process improvement.
3. Provide a vehicle for assessing tester competence.
4. Provide testers with a means of tracking the quality of the system under test.

**28. Consider the following techniques. Which are static and which are dynamic techniques?**

1. Equivalence Partitioning.
2. Use Case Testing.
3. Data Flow Analysis.
4. Exploratory Testing.
5. Decision Testing.
6. Inspections.

Data Flow Analysis and Inspections are static; Equivalence Partitioning, Use Case Testing, Exploratory Testing and Decision Testing are dynamic.

**29. Why are static testing and dynamic testing described as complementary?**

Because they share the aim of identifying defects but differ in the types of defect they find.

**30. What are the phases of a formal review?**

In contrast to informal reviews, formal reviews follow a formal process. A typical formal review process consists of six main steps:

1. Planning
2. Kick-off
3. Preparation
4. Review meeting
5. Rework
6. Follow-up.

**31. What is the role of moderator in review process?**

The moderator (or review leader) leads the review process. He or she determines, in co-operation with the author, the type of review, approach and the composition of the review team. The moderator performs the entry

check and the follow-up on the rework, in order to control the quality of the input and output of the review process. The moderator also schedules the meeting, disseminates documents before the meeting, coaches other team members, paces the meeting, leads possible discussions and stores the data that is collected.

Learn More about Review process in Video Tutorial [here](#)

**32. What is an equivalence partition (also known as an equivalence class)?**

An input or output ranges of values such that only one value in the range becomes a test case.

**33. When should configuration management procedures be implemented?**

During test planning.

**34. A Type of functional Testing, which investigates the functions relating to detection of threats, such as virus from malicious outsiders?**

Security Testing

**35. Testing where in we subject the target of the test , to varying workloads to measure and evaluate the performance behaviours and ability of the target and of the test to continue to function properly under these different workloads?**

Load Testing

**36. Testing activity which is performed to expose defects in the interfaces and in the interaction between integrated components is?**

Integration Level Testing

**37. What are the Structure-based (white-box) testing techniques?**

Structure-based testing techniques (which are also dynamic rather than static) use the internal structure of the software to derive test cases. They are commonly called 'white-box' or 'glass-box' techniques (implying you can see into the system) since they require knowledge of how the software is implemented, that is, how it works. For example, a structural technique may be concerned with exercising loops in the software. Different test cases may be derived to exercise the loop once, twice, and many times. This may be done regardless of the functionality of the software.

**38. When “Regression Testing” should be performed?**

After the software has changed or when the environment has changed Regression testing should be performed.

### **39. What is negative and positive testing?**

A negative test is when you put in an invalid input and receives errors. While a positive testing, is when you put in a valid input and expect some action to be completed in accordance with the specification.

### **40. What is the purpose of a test completion criterion?**

The purpose of test completion criterion is to determine when to stop testing

### **41. What can static analysis NOT find?**

For example memory leaks.

### **42. What is the difference between re-testing and regression testing?**

Re-testing ensures the original fault has been removed; regression testing looks for unexpected side effects.

### **43. What are the Experience-based testing techniques?**

In experience-based techniques, people's knowledge, skills and background are a prime contributor to the test conditions and test cases. The experience of both technical and business people is important, as they bring different perspectives to the test analysis and design process. Due to previous experience with similar systems, they may have insights into what could go wrong, which is very useful for testing.

### **44. What type of review requires formal entry and exit criteria, including metrics?**

Inspection

### **45. Could reviews or inspections be considered part of testing?**

Yes, because both help detect faults and improve quality.

### **46. An input field takes the year of birth between 1900 and 2004 what are the boundary values for testing this field?**

1899,1900,2004,2005

### **47. Which of the following tools would be involved in the automation of regression test? a. Data tester b. Boundary tester c. Capture/Playback d. Output comparator.**

d. Output comparator

### **48. To test a function, what has to write a programmer, which calls the function to be tested and passes it test data.**

Driver

**49. What is the one Key reason why developers have difficulty testing their own work?**

Lack of Objectivity

**50.“How much testing is enough?”**

The answer depends on the risk for your industry, contract and special requirements.

**51. When should testing be stopped?**

It depends on the risks for the system being tested. There are some criteria bases on which you can stop testing.

1. Deadlines (Testing, Release)
2. Test budget has been depleted
3. Bug rate fall below certain level
4. Test cases completed with certain percentage passed
5. Alpha or beta periods for testing ends
6. Coverage of code, functionality or requirements are met to a specified point

**52. Which of the following is the main purpose of the integration strategy for integration testing in the small?**

The main purpose of the integration strategy is to specify which modules to combine when and how many at once.

**53.What are semi-random test cases?**

Semi-random test cases are nothing but when we perform random test cases and do equivalence partitioning to those test cases, it removes redundant test cases, thus giving us semi-random test cases.

**54. Given the following code, which statement is true about the minimum number of test cases required for full statement and branch coverage?**

**Read p**

**Read q**

**IF p+q> 100**

**THEN Print "Large"**

**ENDIF**

**IF p > 50**

**THEN Print "p Large"**

**ENDIF**

1 test for statement coverage, 2 for branch coverage

**55. What is black box testing? What are the different black box testing techniques?**

Black box testing is the software testing method which is used to test the software without knowing the internal structure of code or program. This testing is usually done to check the functionality of an application. The different black box testing techniques are

1. Equivalence Partitioning
2. Boundary value analysis
3. Cause effect graphing

**56. Which review is normally used to evaluate a product to determine its suitability for intended use and to identify discrepancies?**

Technical Review.

**57. Why we use decision tables?**

The techniques of equivalence partitioning and boundary value analysis are often applied to specific situations or inputs. However, if different combinations of inputs result in different actions being taken, this can be more difficult to show using equivalence partitioning and boundary value analysis, which tend to be more focused on the user interface. The other two specification-based techniques, decision tables and state transition testing are more focused on business logic or business rules. A decision table is a good way to deal with combinations of things (e.g. inputs). This technique is sometimes also referred to as a 'cause-effect' table. The reason for this is that there is an associated logic diagramming technique called 'cause-effect graphing' which was sometimes used to help derive the decision table

**58. Faults found should be originally documented by whom?**

By testers.

**59. Which is the current formal world-wide recognized documentation standard?**

There isn't one.

**60. Which of the following is the review participant who has created the item to be reviewed?**

Author

**61. A number of critical bugs are fixed in software. All the bugs are in one module, related to reports. The test manager decides to do regression testing only on the reports module.**

Regression testing should be done on other modules as well because fixing one module may affect other modules.

**62. Why does the boundary value analysis provide good test cases?**

Because errors are frequently made during programming of the different cases near the ‘edges’ of the range of values.

**63. What makes an inspection different from other review types?**

It is led by a trained leader, uses formal entry and exit criteria and checklists.

**64. Why can be tester dependent on configuration management?**

Because configuration management assures that we know the exact version of the testware and the test object.

**65. What is a V-Model?**

A software development model that illustrates how testing activities integrate with software development phases

**66. What is maintenance testing?**

Triggered by modifications, migration or retirement of existing software

**67. What is test coverage?**

Test coverage measures in some specific way the amount of testing performed by a set of tests (derived in some other way, e.g. using specification-based techniques). Wherever we can count things and can tell whether or not each of those things has been tested by some test, then we can measure coverage.

**68. Why is incremental integration preferred over “big bang” integration?**

Because incremental integration has better early defects screening and isolation ability

**69. When do we prepare RTM (Requirement traceability matrix), is it before test case designing or after test case designing?**

It would be before test case designing. Requirements should already be traceable from Review activities since you should have traceability in the Test Plan already. This question also would depend on the organisation. If the organisations do test after development started then requirements must be already traceable to their source. To make life simpler use a tool to manage requirements.

**70. What is called the process starting with the terminal modules?**

Bottom-up integration

**71. During which test activity could faults be found most cost effectively?**

During test planning

**72. The purpose of requirement phase is**

To freeze requirements, to understand user needs, to define the scope of testing

**73. Why we split testing into distinct stages?**

We split testing into distinct stages because of following reasons,

1. Each test stage has a different purpose
2. It is easier to manage testing in stages
3. We can run different test into different environments
4. Performance and quality of the testing is improved using phased testing

**74. What is DRE?**

To measure test effectiveness a powerful metric is used to measure test effectiveness known as DRE (Defect Removal Efficiency) From this metric we would know how many bugs we have found from the set of test cases. Formula for calculating DRE is

DRE=Number of bugs while testing / number of bugs while testing + number of bugs found by user

**75. Which of the following is likely to benefit most from the use of test tools providing test capture and replay facilities? a) Regression testing b) Integration testing c) System testing d) User acceptance testing**

Regression testing

**76. How would you estimate the amount of re-testing likely to be required?**

Metrics from previous similar projects and discussions with the development team

**77. What studies data flow analysis?**

The use of data on paths through the code.

**78. What is Alpha testing?**

Pre-release testing by end user representatives at the developer's site.

## **79. What is a failure?**

Failure is a departure from specified behaviour.

## **80. What are Test comparators?**

Is it really a test if you put some inputs into some software, but never look to see whether the software produces the correct result? The essence of testing is to check whether the software produces the correct result, and to do that, we must compare what the software produces to what it should produce. A test comparator helps to automate aspects of that comparison.

## **81. Who is responsible for document all the issues, problems and open point that were identified during the review meeting**

Scribe

## **82. What is the main purpose of Informal review**

Inexpensive way to get some benefit

## **83. What is the purpose of test design technique?**

Identifying test conditions and Identifying test cases

## **84. When testing a grade calculation system, a tester determines that all scores from 90 to 100 will yield a grade of A, but scores below 90 will not. This analysis is known as:**

Equivalence partitioning

## **85. A test manager wants to use the resources available for the automated testing of a web application. The best choice is Tester, test automater, web specialist, DBA**

## **86. During the testing of a module tester ‘X’ finds a bug and assigned it to developer. But developer rejects the same, saying that it’s not a bug. What ‘X’ should do?**

Send to the detailed information of the bug encountered and check the reproducibility

## **87. A type of integration testing in which software elements, hardware elements, or both are combined all at once into a component or an overall system, rather than in stages.**

Big-Bang Testing

**88. In practice, which Life Cycle model may have more, fewer or different levels of development and testing, depending on the project and the software product. For example, there may be component integration testing after component testing, and system integration testing after system testing.**

V-Model

**89. Which technique can be used to achieve input and output coverage? It can be applied to human input, input via interfaces to a system, or interface parameters in integration testing.**

Equivalence partitioning

**90. “This life cycle model is basically driven by schedule and budget risks” This statement is best suited for...**

V-Model

**91. In which order should tests be run?**

The most important one must tests first

**92. The later in the development life cycle a fault is discovered, the more expensive it is to fix. Why?**

The fault has been built into more documentation, code, tests, etc

**93. What is Coverage measurement?**

It is a partial measure of test thoroughness.

**94. What is Boundary value testing?**

Test boundary conditions on, below and above the edges of input and output equivalence classes. For instance, let say a bank application where you can withdraw maximum Rs.20,000 and a minimum of Rs.100, so in boundary value testing we test only the exact boundaries, rather than hitting in the middle. That means we test above the maximum limit and below the minimum limit.

**95. What is Fault Masking?**

Error condition hiding another error condition.

**96. What does COTS represent?**

Commercial off The Shelf.

**97. The purpose of which is allow specific tests to be carried out on a system or network that resembles as closely as possible the environment where the item under test will be used upon release?**

Test Environment

## **98. What can be thought of as being based on the project plan, but with greater amounts of detail?**

Phase Test Plan

## **99. What is exploratory testing?**

Exploratory testing is a hands-on approach in which testers are involved in minimum planning and maximum test execution. The planning involves the creation of a test charter, a short declaration of the scope of a short (1 to 2 hour) time-boxed test effort, the objectives and possible approaches to be used. The test design and test execution activities are performed in parallel typically without formally documenting the test conditions, test cases or test scripts. This does not mean that other, more formal testing techniques will not be used. For example, the tester may decide to use boundary value analysis but will think through and test the most important boundary values without necessarily writing them down. Some notes will be written during the exploratory-testing session, so that a report can be produced afterwards.

## **100. What is “use case testing”?**

In order to identify and execute the functional requirement of an application from start to finish “use case” is used and the techniques used to do this is known as “Use Case Testing”

**Bonus!**

## **101. What is the difference between STLC ( Software Testing Life Cycle) and SDLC ( Software Development Life Cycle) ?**

The complete **Verification and Validation of software** is done in **SDLC**, while **STLC only does Validation** of the system. SDLC is a part of STLC.

## **102. What is traceability matrix?**

The relationship between test cases and requirements is shown with the help of a document. This document is known as traceability matrix.

## **103. What is Equivalence partitioning testing?**

Equivalence partitioning testing is a software testing technique which divides the application input test data into each partition at least once of equivalent data from which test cases can be derived. By this testing method it reduces the time required for software testing.

#### **104. What is white box testing and list the types of white box testing?**

White box testing technique involves selection of test cases based on an analysis of the internal structure (Code coverage, branches coverage, paths coverage, condition coverage etc.) of a component or system. It is also known as Code-Based testing or Structural testing. Different types of white box testing are

1. Statement Coverage
2. Decision Coverage

#### **105. In white box testing what do you verify?**

In white box testing following steps are verified.

1. Verify the security holes in the code
2. Verify the incomplete or broken paths in the code
3. Verify the flow of structure according to the document specification
4. Verify the expected outputs
5. Verify all conditional loops in the code to check the complete functionality of the application
6. Verify the line by line coding and cover 100% testing

#### **106. What is the difference between static and dynamic testing?**

Static testing: During Static testing method, the code is not executed and it is performed using the software documentation.

Dynamic testing: To perform this testing the code is required to be in an executable form.

#### **107. What is verification and validation?**

Verification is a process of evaluating software at development phase and to decide whether the product of a given application satisfies the specified requirements. Validation is the process of evaluating software at the end of the development process and to check whether it meets the customer requirements.

#### **108. What are different test levels?**

There are four test levels

1. Unit/component/program/module testing
2. Integration testing
3. System testing
4. Acceptance testing

## **109. What is Integration testing?**

Integration testing is a level of software testing process, where individual units of an application are combined and tested. It is usually performed after unit and functional testing.

## **110. What are the tables in testplans?**

Test design, scope, test strategies , approach are various details that Test plan document consists of.

1. Test case identifier
2. Scope
3. Features to be tested
4. Features not to be tested
5. Test strategy & Test approach
6. Test deliverables
7. Responsibilities
8. Staffing and training
9. Risk and Contingencies

## **111. What is the difference between UAT (User Acceptance Testing) and System testing?**

**System Testing:** System testing is finding defects when the system under goes testing as a whole, it is also known as end to end testing. In such type of testing, the application undergoes from beginning till the end.

**UAT:** User Acceptance Testing (UAT) involves running a product through a series of specific tests which determines whether the product will meet the needs of its users.

# SOFTWARE ENGINEERING INTERVIEW QUESTIONS

[http://www.tutorialspoint.com/software\\_engineering/software\\_engineering\\_interview\\_questions.htm](http://www.tutorialspoint.com/software_engineering/software_engineering_interview_questions.htm)  
Copyright © tutorialspoint.com

Dear readers, these **Software Engineering Interview Questions** have been designed especially to get you acquainted with the nature of questions you may encounter during your interview for the subject of **Software Engineering**. As per my experience, good interviewers hardly planned to ask any particular question during your interview, normally questions start with some basic concept of the subject and later they continue based on further discussion and what you answer:

## Q.What is computer software?

A. Computer software is a **complete package**, which includes **software program**, its **documentation and user guide on how to use the software**.

Yes, by looking on the project documentation, we are going to decide as I attend the design meeting with the architect and I also used to give some suggestions about how to achieve this functionality to fulfill this requirement.

## Q.Can you differentiate computer software and computer program?

Architect will decide X framework for the project, but as a developer, I used to evaluate whether this framework will suitable for the project or not. If we find any drawbacks then we discuss

A. A computer program is **piece of programming code** which performs a **well defined task** where as software **includes programming code, its documentation and user guide**.

## Q.What is software engineering?

A. Software engineering is an **engineering branch** associated with software system development.

## Q.When you know programming, what is the need to learn software engineering concepts?

A. A person who knows how to build a wall may not be good at building an entire house. Likewise, a person who can write programs may not have knowledge of other concepts of Software Engineering. The software engineering concepts guide programmers on how to assess requirements of end user, design the algorithms before actual coding starts, create programs by coding, testing the code and its documentation.

## Q.What is software process or Software Development Life Cycle (SDLC)?

A. Software Development Life Cycle, or software process is the **systematic development of software by following every stage in the development process namely, Requirement Gathering, System Analysis, Design, Coding, Testing, Maintenance and Documentation** in that order.

## Q.What are SDLC models available?

A. There are several SDLC models available such as **Waterfall Model, Iterative Model, Spiral model, V-model and Big-bang Model** etc.

## **Q.What are various phases of SDLC?**

**A.** The generic phases of SDLC are: Requirement Gathering, System Analysis and Design, Coding, Testing and implementation. The phases depend upon the model we choose to develop software.

## **Q.Which SDLC model is the best?**

**A.** SDLC Models are adopted as per requirements of development process. It may vary software-to-software to ensuring which model is suitable.

We can select the best SDLC model if following answers are satisfied -

- Is SDLC suitable for selected technology to implement the software ?
- Is SDLC appropriate for client's requirements and priorities ?
- Is SDLC model suitable for size and complexity of the software ?
- Is the SDLC model suitable for type of projects and engineering we do ?
- Is the SDLC appropriate for the geographically co-located or dispersed developers ?

## **Q.What is software project management?**

**A.** Software project management is process of managing all activities like time, cost and quality management involved in software development.

## **Q.Who is software project manager?**

**A.** A software project manager is a person who undertakes the responsibility of carrying out the software project.

## **Q.What does software project manager do?**

**A.** Software project manager is engaged with software management activities. He is responsible for project planning, monitoring the progress, communication among stakeholders, managing risks and resources, smooth execution of development and delivering the project within time, cost and quality constraints.

## **Q.What is software scope?**

**A.** Software scope is a well-defined boundary, which encompasses all the activities that are done to develop and deliver the software product.

The software scope clearly defines all functionalities and artifacts to be delivered as a part of the software. The scope identifies what the product will do and what it will not do, what the end product will contain and what it will not contain.

## **Q.What is project estimation?**

**A.** It is a process to estimate various aspects of software product in order to calculate the cost of development in terms of efforts, time and resources. This estimation can be derived from past experience, by consulting experts or by using pre-defined formulas.

**Q.How can we derive the size of software product?**

**A.** Size of software product can be calculated using either of two methods -

- Counting the lines of delivered code
- Counting delivered function points

**Q.What are function points?**

**A.** Function points are the various features provided by the software product. It is considered as a unit of measurement for software size.

**Q.What are software project estimation techniques available?**

**A.** There are many estimation techniques available. The most widely used are -

- Decomposition technique (Counting Lines of Code and Function Points)
- Empirical technique (Putnam and COCOMO).

**Q.What is baseline?**

**A.** Baseline is a measurement that defines completeness of a phase. After all activities associated with a particular phase are accomplished, the phase is complete and acts as a baseline for next phase.

**Q.What is Software configuration management?**

**A.** Software Configuration management is a process of tracking and controlling the changes in software in terms of the requirements, design, functions and development of the product.

**Q.What is change control?**

**A.** Change control is function of configuration management, which ensures that all changes made to software system are consistent and made as per organizational rules and regulations.

**Q.How can you measure project execution?**

**A.** We can measure project execution by means of Activity Monitoring, Status Reports and Milestone Checklists.

**Q.Mention some project management tools.**

**A.** There are various project management tools used as per the requirements of software project and organization policies. They include Gantt Chart, PERT Chart,

Resource Histogram, Critical Path Analysis, Status Reports, Milestone Checklists etc.

## Q.What are software requirements?

A. Software requirements are functional description of proposed software system. Requirements are assumed to be the description of target system, its functionalities and features. Requirements convey the expectations of users from the system.

## Q.What is feasibility study?

A. It is a measure to assess how practical and beneficial the software project development will be for an organization. The software analyzer conducts a thorough study to understand economic, technical and operational feasibility of the project.

- **Economic** - Resource transportation, cost for training, cost of additional utilities and tools and overall estimation of costs and benefits of the project.
- **Technical** - Is it possible to develop this system ? Assessing suitability of machine(s) and operating system(s) on which software will execute, existing developers' knowledge and skills, training, utilities or tools for project.
- **Operational** - Can the organization adjust smoothly to the changes done as per the demand of project ? Is the problem worth solving ?

## Q.How can you gather requirements?

A. Requirements can be gathered from users via interviews, surveys, task analysis, brainstorming, domain analysis, prototyping, studying existing usable version of software, and by observation.

## Q.What is SRS?

A. SRS or Software Requirement Specification is a document produced at the time of requirement gathering process. It can be also seen as a process of refining requirements and documenting them.

## Q.What are functional requirements?

A. Functional requirements are functional features and specifications expected by users from the proposed software product.

## Q.What are non-functional requirements?

A. Non-functional requirements are implicit and are related to security, performance, look and feel of user interface, interoperability, cost etc.

## Q.What is software measure?

A. Software Measures can be understood as a process of quantifying and symbolizing various attributes and aspects of software.

## **Q.What is software metric?**

**A.** Software Metrics provide measures for various aspects of software process and software product. They are divided into –

- **Requirement metrics** : Length requirements, completeness
- **Product metrics** : Lines of Code, Object oriented metrics, design and test metrics
- **Process metrics**: Evaluate and track budget, schedule, human resource.

## **Q.What is modularization?**

**A.** Modularization is a technique to divide a software system into multiple discrete modules, which are expected to carry out task(s) independently.

## **Q.What is concurrency and how it is achieved in software?**

**A.** Concurrency is the tendency of events or actions to happen simultaneously. In software, when two or more processes execute simultaneously, they are called concurrent processes.

## **Example**

While you initiate print command and printing starts, you can open a new application.

Concurrency, is implemented by splitting the software into multiple independent units of execution namely processes and threads, and executing them in parallel.

## **Q.What is cohesion?**

**A.** Cohesion is a measure that defines the degree of intra-dependability among the elements of the module.

## **Q.What is coupling?**

**A.** Coupling is a measure that defines the level of inter-dependability among modules of a program.

## **Q.Mentions some software analysis & design tools?**

**A.** These can be: **DFDs (Data Flow Diagrams)**, **Structured Charts**, Structured English, Data Dictionary, HIPO (Hierarchical Input Process Output) diagrams, **ER (Entity Relationship) Diagrams** and Decision tables.

## **Q.What is level-0 DFD?**

**A.** Highest abstraction level DFD is known as Level 0 DFD also called a context level DFD, which depicts the entire information system as one diagram concealing all the underlying details.

## **Q.What is the difference between structured English and Pseudo Code?**

A. Structured English is native English language used to write the structure of a program module by using programming language keywords, whereas, Pseudo Code is more close to programming language and uses native English language words or sentences to write parts of code.

## **Q.What is data dictionary?**

A. Data dictionary is referred to as meta-data. Meaning, it is a repository of data about data. Data dictionary is used to organize the names and their references used in system such as objects and files along with their naming conventions.

## **Q.What is structured design?**

A. Structured design is a conceptualization of problem into several well-organized elements of solution. It is concern with the solution design and based on 'divide and conquer' strategy.

## **Q.What is the difference between function oriented and object oriented design?**

A. Function-oriented design is comprised of many smaller sub-systems known as functions. Each function is capable of performing significant task in the system. Object oriented design works around the real world objects (entities), their classes (categories) and methods operating on objects (functions).

## **Q.Briefly define top-down and bottom-up design model.**

A. Top-down model starts with generalized view of system and decomposes it to more specific ones, whereas bottom-up model starts with most specific and basic components first and keeps composing the components to get higher level of abstraction.

## **Q.What is the basis of Halstead's complexity measure?**

A. Halstead's complexity measure depends up on the actual implementation of the program and it considers tokens used in the program as basis of measure.

## **Q.Mention the formula to calculate Cyclomatic complexity of a program?**

A. Cyclomatic complexity uses graph theory's formula:  $V(G) = e - n + 2$

## **Q.What is functional programming?**

A. Functional programming is style of programming language, which uses the concepts of mathematical function. It provides means of computation as mathematical functions, which produces results irrespective of program state.

## **Q.Differentiate validation and verification?**

**A.** Validation checks if the product is made as per user requirements whereas verification checks if proper steps are followed to develop the product.

Validation confirms the right product and verification confirms if the product is built in a right way.

### **Q.What is black-box and white-box testing?**

**A.** Black-box testing checks if the desired outputs are produced when valid input values are given. It does not verify the actual implementation of the program.

White-box testing not only checks for desired and valid output when valid input is provided but also it checks if the code is implemented correctly.

Criteria	Black Box Testing	White Box Testing
Knowledge of software program, design and structure essential	No	Yes
Knowledge of Software Implementation essential	No	Yes
Who conducts this test on software	Software Testing Employee	Software Developer
baseline reference for tester	Requirements specifications	Design and structure details

### **Q.Quality assurance vs. Quality Control?**

**A.** Quality Assurance monitors to check if proper process is followed while software developing the software.

Quality Control deals with maintaining the quality of software product.

### **Q.What are various types of software maintenance?**

**A.** Maintenance types are: corrective, adaptive, perfective and preventive.

- **Corrective**

Removing errors spotted by users

- **Adaptive**

tackling the changes in the hardware and software environment where the software works

- **Perfective maintenance**

implementing changes in existing or new requirements of user

- **Preventive maintenance**

taking appropriate measures to avoid future problems

## **Q.What is software re-engineering?**

**A.** Software re-engineering is process to upgrade the technology on which the software is built without changing the functionality of the software. This is done in order to keep the software tuned with the latest technology.

## **Q.What are CASE tools?**

**A.** CASE stands for Computer Aided Software Engineering. CASE tools are set of automated software application programs, which are used to support, accelerate and smoothen the SDLC activities.

## **What is Next?**

Further, you can go through your past assignments you have done with the subject and make sure you are able to speak confidently on them. If you are fresher then interviewer does not expect you will answer very complex questions, rather you have to make your basics concepts very strong.

Second it really doesn't matter much if you could not answer few questions but it matters that whatever you answered, you must have answered with confidence. So just feel confident during your interview. We at tutorialspoint wish you best luck to have a good interviewer and all the very best for your future endeavor. Cheers :-)

## OOPS – Object Oriented Programming

- Data Abstraction
- Encapsulation
- Inheritance
- Polymorphism

<u>Overloading</u>	<u>Overriding</u>
Same method name, different method signature	Same method name, same method signature But method body/impl will vary
Within the same class	Between class and its sub class
Return types <b>may vary</b>	Return types are same
Declared exceptions <b>may vary</b>	Declared exceptions will also be same

<u>Class</u>	<u>Abstract class</u>	<u>Interface</u>
Implements(provide method body) all of its methods	Implements some, none or all of its methods	Implements none of its methods
Can have instances	Cannot have instances	Cannot have instances
Can have sub classes (extends)	Must have sub class	Cannot have sub class but may have class that (implements) this interface

### Singleton Class :

- Should have a private constructor and at least one constructor
- Should have a private static object of the class
- Should have public method to return that private static instance

only static methods can access static fields.

```
public class Single {
 private static Single s;
 private Single() {
 }
 public static Single giveMeInstance() {
 if (null == s) {
 s = new Single(); // once ever
 }
 return s;
 }
}
```

<u>Vector</u> round robin technique / priority wise tech	<u>ArrayList</u>
Synchronized <b>2 different threads accessing at same time</b>	Unsynchronized
Allows duplicates	Allows null
<u>Hashtable</u>	<u>HashMap</u>
Synchronized	Unsynchronized
No duplicates	Allows null
<u>ArrayList</u>	<u>LinkedList</u>
Slow access	Fast access
Random access	No random access

	<u>Ordered</u>	<u>Duplicates</u>
LIST	Yes	Yes
SET	No	No

	ArrayList	Vector	LinkedList	HashMap	LinkedHashMap	Hashtable	TreeMap	HashSet	LinkedHashSet	TreeSet
Allows Null?	Yes	Yes	Yes	Yes ( But one key and multiple values )	Yes ( But one key and multiple values )	No	Yes ( But zero keys and multiple values )	Yes	Yes	No
Allows Duplicates?	Yes	Yes	Yes	No	No	No	No	No	No	No
Retrieves Sorted results?	No	No	No	No	No	No	Yes	No	No	Yes
Retrieves same as Insertion order?	Yes	Yes	Yes	No	Yes	No	No	No	Yes	No

LIST – ordered

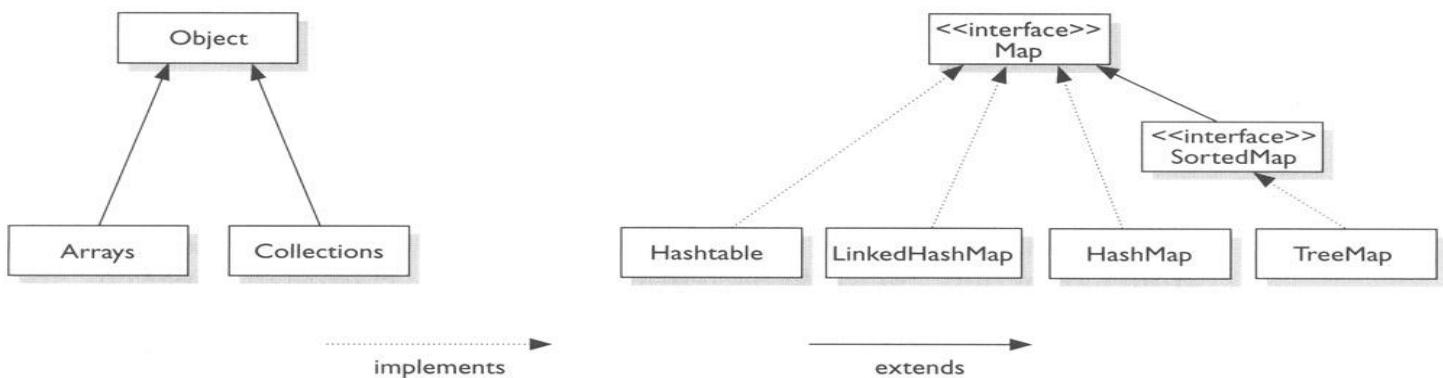
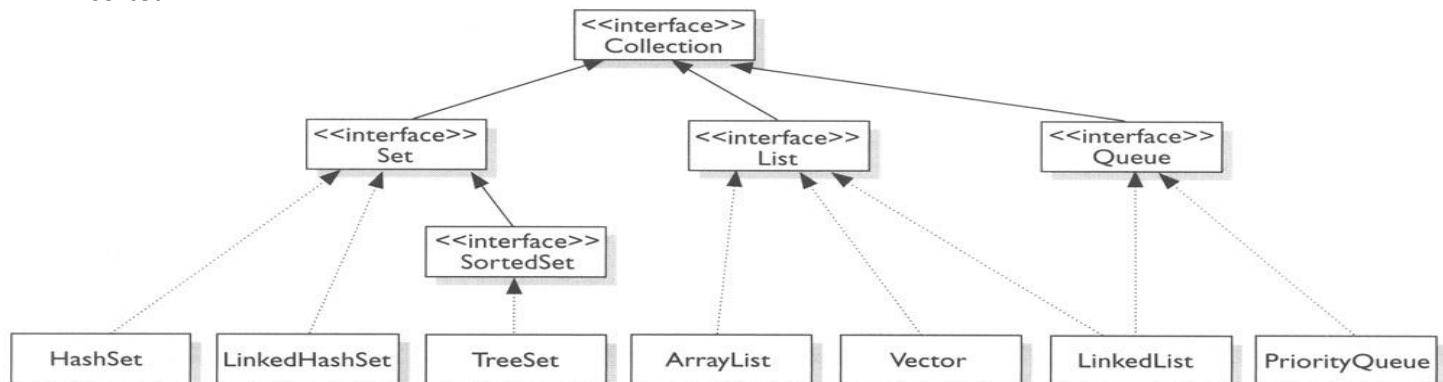
LINKED – faster access, and ordered access

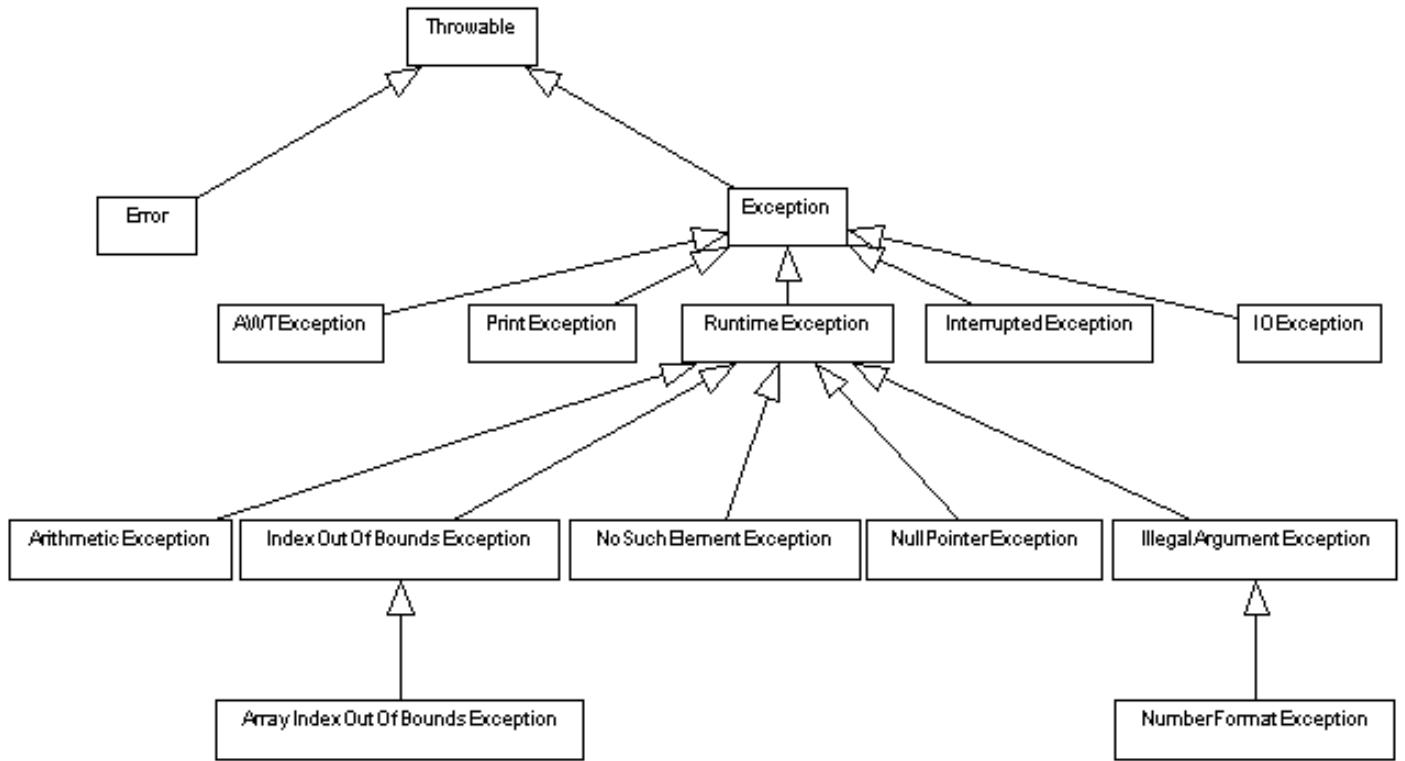
SET and MAP – unique values, no duplicates

MAP – key-value pairs, and no duplicates

HASH – not sorted

TREE – sorted





Servlet extends GenericServlet

- `init()`
- `service()`
- `destroy()`

Servlet extends HttpServlet

- `init()`
- `service()`
- `destroy()`
- `doGet()`
- `doPost()`

<u>ServletConfig</u>	<u>ServletContext</u>
Specific to one servlet	Applies to all servlets
Used to pass initialization parameters to the servlet	Used to define context parameters common to all servlets
<init-param> tags are used	<context-param> tags are used

### Servlet Life Cycle

If an instance of the servlet does not exist, the web container

- a. Loads the servlet class.
- b. Creates an instance of the servlet class.
- c. Initializes the servlet instance by calling the `init` method.
- d. Invokes the `service` method, passing request and response objects.

## JSP Life Cycle

When a request is mapped to a JSP page, the web container first checks whether the JSP page's servlet is older than the JSP page. If the servlet is older, the web container **translates** the JSP page into a servlet class and **compiles** the class. After the page has been translated and compiled, the JSP page's servlet (for the most part) follows the servlet life cycle

Directives	<%@ %>
Declarations	<%! %>
Expressions	<%= %>
Scriptlets	<% %>

### Directives :

<%@ page import = "" %>
<%@ include file = "" %>
<%@ taglib uri = "" %>
<%@ page errorPage = "" %>

### REQUEST DISPATCHER :

<jsp:include page = "" />	response of the second jsp is included in the response of the first jsp
<jsp:forward page = "" />	request is routed to several other jsps before response is sent back
<jsp:param name = "" value = "" />	Parameters you pass along, while forwarding

response.forward("other.jsp") – **same** request is passed to other jsp and control does not go to UI  
 response.sendRedirect("other.jsp") – control goes to UI and then gets re-routed to other jsp with **new** request

### ACID :

Transaction	Set of statements executed on a database or file system
Atomicity	Set of statements executed as a single unit of work
Consistency	Txn exists in a consistent state before and after the execution
Isolation	One txn executes independent of the other txn
Durability	Once COMMIT is fired txn survives even network failures

Dirty Read	Reading uncommitted data. First txn could roll back before committing data read by second txn		
Non-Repeatable Read	In a same transaction same query yields different results. This happens when another transaction updates the data returned by other transaction.		
Phantom Read	First txn still not done adding data, and may add more data after second txn made a read		

ISOLATION LEVEL	Dirty Read	Non Repeatable Read	Phantom Read
READ_COMMITTED	YES	NO	NO
REPEATABLE_READ	YES	YES	NO
SERIALIZABLE	YES	YES	YES

Normalization - It is the process of organizing data in order to achieve 2 things :

1. Eliminate redundant data
2. Ensure data dependency makes sense by store related data in the same table

Data Definition Language (DDL)	create, alter, rename, drop, truncate
Data Manipulation Language (DML)	select, insert, update, delete (CRUD)
Data Control Language (DCL)	grant, revoke
Transaction Control Language (TCL)	commit, rollback, savepoint

<u>Delete</u>	<u>Truncate</u>	<u>Drop</u>
Delete some or all records	Delete all records	Remove table
WHERE clause can be used	WHERE clause cannot be used	
Need explicit COMMIT	Auto COMMIT	
Can be rolled back	Cannot be rolled back	
Does not trim the table	Trims the table	

<u>Stored Procedure</u>	<u>Function</u>	<u>Trigger</u>
May or may not return a value	Must return a value	Code fragment that runs before or after a row or table is modified
Called by EXEC command	Called from SQL statements	
Can call function	Cannot call procedure	

Join	Returns rows when there is atleast one match in both tables
Left	All rows of left table with or without matching rows in the right table
Left Outer	Rows of left without matching rows in right table
Right	All rows of right table with or without matching rows in the left table
Right Outer	Rows of right without matching rows in left table
Inner	All rows common between left and right tables
Outer	All rows from both left and right tables except the common/matching ones

<u>Clustered Index</u>	<u>Non-clustered Index</u>
Order of the indexes matches order of data storage	Order of the indexes does not match order of data storage
Telephone directory	Index at the back of the book
Sorting of the indexes sorts the data	Sorting of the indexes does not sort the data

<u>Optimistic locking</u>	<u>Pessimistic locking</u>
Assuming data duplication will never occur, bother about locks and eliminate duplicates as they occur	Assuming data duplication always occurs, address duplication before it occurs

Performance tuning can be done in the following ways :

- Too much normalization is bad, denormalize where required
- Too much of indexing is bad, optimize index usage
- Reduce no.of.cols that make up the composite key
- Proper partitioning of table spaces, have separate space for BLOBs and CLOBs
- Use stored procedures

<b><u>View</u></b>	<b><u>Materialized View</u></b>
Runs the query each time it is accessed	Runs the query only once and updates periodically
Gets latest data but performance depends on well-formed query	Fast but may not refresh on time to get latest data

Only when there is a IS-A relation, use Inheritance. If you want just code reuse, use Composition.  
Use Interfaces if you want polymorphism.

Aggregation	Contained object is not dependent on container object, when container is destroyed contained remains undestroyed. E.g. School-students
Composition	Contained object is dependent on container object, when container is destroyed contained is also destroyed. E.g. House-rooms

#### **Data Modeling :**

- Identify entities
- Identify entity attributes
- Apply naming conventions as per logical model standards
- Identify relationships – cardinality
- Apply design patterns
- Assign keys
- Normalize : eliminate data redundancy
- Denormalize : improve performance

#### **CREATIONAL Design Patterns :**

Factory	Abstraction or interface, to let sub class or the implementing class, to decide, which class is to be instantiated, or which method is to be called
Prototype	Cloning of an object to reduce the cost of creation. E.g. method overriding
Singleton	Only one instance of the object exists at anytime

#### **STRUCTURAL Design Patterns :**

Adapter	Convert existing interface to new interface in order to achieve reusability of unrelated classes. E.g. wrapper classes
Composite	Build complex objects out of elemental objects. E.g. file directory structure
Decorator	Add additional functions to existing interface dynamically. E.g. JScrollPane to JTextArea
Façade	Make complex system look simple by providing a generic interface
Proxy	Simple object to represent complex object, providing a placeholder for the complex object to access it. E.g. stub-skeleton

#### **BEHAVIORAL Design Patterns :**

Chain of Responsibility	Let more than one object participate in handling the request without the knowledge of each other. E.g. servlet chaining
Observer	One object changes state and all dependent objects are updated automatically. E.g. wait-notify

#### **J2EE Design Patterns :**

Front Controller	Single component to receive all input requests
Service Locator	Centralizing the distributed object lookup
Business Delegate	Decoupling the presentation and service tiers
DAO	Uniform interface to access multiple databases

TO	Serialized object to act as data carrier for related attributes

**STRUTS :**Open source framework built on MVC architecture for developing web applications using Java EE.

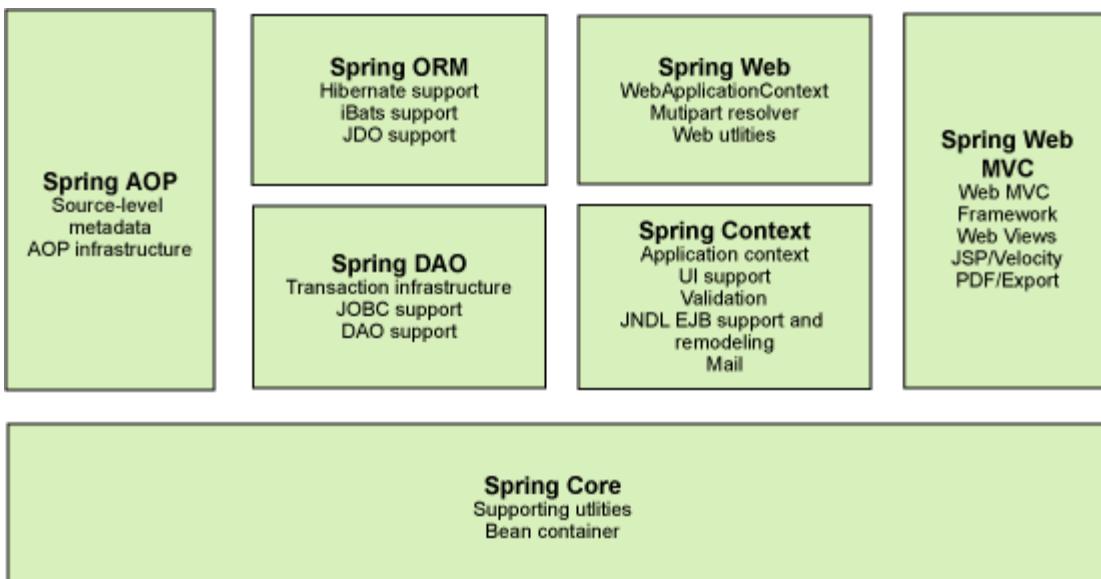
Model	Java beans, EJB
View	HTML, JSP
Controller	ActionServlet
ActionForm	Java beans representing form inputs containing request parameters from the view, referencing the action bean. validate() – used to validate the input parameters, called before form bean is handed over to the Action class, it returns ActionErrors object. ActionErrors validate(ActionMapping, HttpServletRequest)
ActionMapping	Contains all maps between, views to other views, and views to action classes
Deployment descriptor	Struts-config.xml , contains : - form beans - action classes - action mappings - global forwards - controller configuration(action servlet)
Action class	Adapter between request contents and logic that need to be executed. execute()- business logic resides here execute(ActionMapping, ActionForm, Request, Response)

#### **Struts Life Cycle : when a req comes from a jsp or html,**

- Retrieve or create form bean
- Store form bean in appropriate scope
- Reset properties/attributes
- Populate properties
- Validate properties
- Pass form bean to action class

#### **SPRING:**

Core	BeanFactory, IOC engine
Context	Config file to define enterprise services (beans.xml)
AOP	
DAO (JDBC)	
ORM (Hibernate, JDO, IBatis)	
Web(MVC: Struts,JSF, Servlets, JSP)	
Web MVC	



BeanFactory	Collection of beans, holds bean definitions, auto-instantiates beans at runtime
ApplicationContext	<p>BeanFactory + resolves text messages, internationalization, loads file resources, event listeners</p> <ul style="list-style-type: none"> <li><b>ClassPathXmlApplicationContext</b> : It Loads context definition from an XML file located in the classpath, treating context definitions as classpath resources. The application context is loaded from the application's classpath by using the code . ApplicationContext context = new ClassPathXmlApplicationContext("bean.xml");</li> <li><b>FileSystemXmlApplicationContext</b> : It loads context definition from an XML file in the filesystem. The application context is loaded from the file system by using the code . ApplicationContext context = new FileSystemXmlApplicationContext("bean.xml");</li> <li><b>XmIWebApplicationContext</b> : It loads context definition from an XML file contained within a web application.</li> </ul>

#### Bean Scopes :

Singleton(d)	Single definition maps to single object instance
Prototype	Single definition maps to multiple object instances
Request	Single definition maps to life cycle of a HttpRequest
Session	Single definition maps to life cycle of a HttpSession

IOC	You do not create an object but define how it is to be created, so that they are automatically instantiated at runtime
Dependency Injection	<p>Do not connect services and components but define the dependency of components on services, so that those services are wired-in at runtime</p> <ul style="list-style-type: none"> <li>- Constructor</li> <li>- Setter</li> <li>- Interface</li> </ul>

#### Spring Life Cycle :

- Container finds bean definition from config file and instantiate the bean
- Using IOC dependency injection , populates all properties as specified in config file
- Call init()
- Call service methods

#### Advantages of Spring :

- POJO programming

- Layered architecture
- Open source (free)
- IOC, DI and AOP

### **Spring-Hibernate Integration :**

2 ways to connect Spring and Hibernate :

- IOC with Hibernate template (there is a Spring JDBC template also, for doing DB txns via JDBC)
- Extend HibernateDAOsupport and AOP
  - o Configure HibernateSessionFactory
  - o Extend the DAO implementation of the HibernateDAOsupport class
  - o Wire in the transaction support using AOP

In essence the Spring configuration files (that can have any name by the way, not just the generic applicationContext.xml) are treated as classpath resources and filed under src/main/resources. During the build process, these are then copied into the WEB-INF/classes directory which is the normal place for these files to end up. Variations include an additional spring directory (e.g. src/main/resources/spring) to separate the Spring contexts from other resources dedicated to application frameworks.

```
<context-param>
<param-name>contextConfigLocation</param-name>
<param-value>/WEB-INF/applicationContext.xml</param-value>
</context-param>
```

In a large project structure, the Spring's bean configuration files are located in different folders for easy maintainability and modular. For example, Spring-Common.xml in common folder, Spring-Connection.xml in connection folder, Spring-ModuleA.xml in ModuleA folder...and etc.

You may load multiple Spring bean configuration files in the code :

```
ApplicationContext context =new ClassPathXmlApplicationContext(newString[]{"Spring-Common.xml","Spring-Connection.xml","Spring-ModuleA.xml"});
```

Put all spring xml files under project classpath.

```
project-classpath/Spring-Common.xml
project-classpath/Spring-Connection.xml
project-classpath/Spring-ModuleA.xml
```

The above ways are lack of organizing and error prone, the better way should be organized all your Spring bean configuration files into a single XML file. For example, create a Spring-All-Module.xml file, and import the entire Spring bean files like this :

File : Spring-All-Module.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

 <import resource="common/Spring-Common.xml"/>
 <import resource="connection/Spring-Connection.xml"/>
 <import resource="moduleA/Spring-ModuleA.xml"/>

</beans>
```

Now you can load a single xml file like this :

```
ApplicationContext context =new ClassPathXmlApplicationContext(Spring-All-Module.xml);
```

Put this file under project classpath.

```
project-classpath/Spring-All-Module.xml
```

AOP	Modularize cross cutting concerns such as logging, transaction management, etc.
Aspect	Cross cutting concern
Joint Point	A certain point in the execution of a program
Advice	Action taken by Aspect at a Joint Point. Can be done in 2 ways : <ol style="list-style-type: none"> <li>Annotations based - it is defined by a 'point cut expression'.</li> <li>Schema based – everything defined using &lt;aop: tags in the config file</li> </ol>

	Before advice – advice executes before joint point After returning – advice executes after joint point exits normally without exception After throwing – advice executes after joint point throws exception After advice (finally) – advice executes after joint point, whether exits normally or with exception Around advice – advice executes both before and after joint point
Interceptors	preHandle() – fired before request is processed postHandle() – fired after request is processed afterCompletion() – fired after view is rendered

**Hibernate:**

- .hbm files
- .cfg file

**Advantages of Hibernate :**

- Boiler plate code is abstracted out into xml files
- Mapping between tables and objects using xml
- Database independent

**Disadvantages of Hibernate :**

- Have to learn new API
- Debug and performance issues
- Slower than JDBC
- Gets complicated when there is many-many relationships

<b><u>Session</u></b>	<b><u>SessionFactory</u></b>
1st level cache	2 <sup>nd</sup> level cache
Data is shared across txns within the same session	Data is shared across sessions

<b><u>Session.load()</u></b>	<b><u>Session.get()</u></b>
If you are certain object exists in cache, avoid DB hit	If you are not sure if object exists in cache, hit the DB to fetch data
If object not found in cache, it throws an exception	If object not found in cache, it returns null, so check for null

Transient – object is instantiated but not associated with session

Persistent – object is associated with session

Detached – object exists but detached from session (session is closed)

<b><u>update()</u></b>	<b><u>merge()</u></b>
Use update() when you are sure that object already exists and is associated to a session	Use merge() when you are not sure that object exists and is associated to a session

**WEB SERVICES**

Reusable component ,that can be exposed as a service, and made available over the network , for remote access.

Interoperability	ability to integrate when we have diverse systems, platforms, languages, frameworks, applications
XML	language gives capability to send data in generic format that can be ported across network over communication protocol like SOAP

SOAP	<p>Simple Object Access Protocol allows communication between applications located on remote machines. It in turn uses transport protocols like HTTP, SMTP, and FTP.</p> <p>Envelope</p> <ul style="list-style-type: none"> <li>- Header(o)</li> <li>- Body           <ul style="list-style-type: none"> <li>o Request</li> <li>o Response</li> <li>o Fault(o)</li> </ul> </li> </ul>
------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### **SOAP Faults**

Version Mismatch	Invalid namespace in envelope
Must understand	SOAP processor is unable to process header
Client	Client incorrectly formed the SOAP message
Server	Server cannot process SOAP message

### **Principles of SOA**

1	Standard service contract between provider and consumer
2	Loose coupling to reduce dependency and maintain only awareness of each other
3	Abstraction to hide logic from outside world
4	Autonomous so that services alone have total control over the logic
5	Reusable
6	Statelessness
7	Granularity
8	Discoverability
9	Composability
10	Interoperability

### **Webservice protocol stack**

Service transport layer	Responsible for transporting messages between applications using HTTP, FTP, and SMTP
XML messaging layer	Responsible for encoding messages into common XML format so that both provider and consumer can understand
Service discovery layer	Responsible for centralized distributed object lookup providing publish/subscribe using UDDI
Service description layer	Responsible for describing the public interface for the service using WSDL

<b>Bottom Up(Java first)</b>	<b>Top Down(WSDL first)</b> – write Java interface, use WSDL annotations(JAXWS) and then generate WSDL
Requires less knowledge of WSDL, XML, XSD	Requires more knowledge of WSDL, XML, XSD
Interoperability issues may occur with consumers/providers written in other languages	Better interoperability
If end point interface changes, WSDL changes , and client to be rewritten	Less sensitive to change

### **WSDL elements :**

Types	Data types, pojos, XSD elements. All XSD types can be declared in separate file and then linked into WSDL
Message	Message types based on XSD types for REQUEST, RESPONSE and FAULT
Port Type	Links operations and message types.

	-request -response -fault
Binding	Protocol and data format for particular port type
Service	Binds java service to port, this is where you define soap address(end point)

```

<definitions name="EndorsementSearch"
 targetNamespace="http://namespaces.snowboard-info.com"
 xmlns:es="http://www.snowboard-info.com/EndorsementSearch.wsdl"
 xmlns:esxsd="http://schemas.snowboard-info.com/EndorsementSearch.xsd"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns="http://schemas.xmlsoap.org/wsdl/"/>

 <!-- omitted types section with content model schema info -->

 <message name="GetEndorsingBoarderRequest">
 <part name="body" element="esxsd:GetEndorsingBoarder"/>
 </message>

 <message name="GetEndorsingBoarderResponse">
 <part name="body" element="esxsd:GetEndorsingBoarderResponse"/>
 </message>

 <portType name="GetEndorsingBoarderPortType">
 <operation name="GetEndorsingBoarder">
 <input message="es:GetEndorsingBoarderRequest"/>
 <output message="es:GetEndorsingBoarderResponse"/>
 <fault message="es:GetEndorsingBoarderFault"/>
 </operation>
 </portType>

 <binding name="EndorsementSearchSoapBinding" type="es:GetEndorsingBoarderPortType">
 <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
 <operation name="GetEndorsingBoarder">
 <soap:operation soapAction="http://www.snowboard-info.com/EndorsementSearch"/>
 <input><soap:body use="literal" namespace="http://schemas.snowboard-info.com/EndorsementSearch.xsd"/></input>
 <output><soap:body use="literal" namespace="http://schemas.snowboard-info.com/EndorsementSearch.xsd"/></output>
 <fault><soap:body use="literal" namespace="http://schemas.snowboard-info.com/EndorsementSearch.xsd"/></fault>
 </operation>
 </binding>

 <service name="EndorsementSearchService">
 <documentation>snowboarding-info.com Endorsement Service</documentation>
 <port name="GetEndorsingBoarderPort" binding="es:EndorsementSearchSoapBinding">
 <soap:address location="http://www.snowboard-info.com/EndorsementSearch"/>
 </port>
 </service>
</definitions>
```

JAXWS specification is used, it is pre-included in JEE5. Once you have application server plugin, if you create a web project, the required jars are auto included by application server.

You use Endpoint.publish("http://...",new MyService()). If you paste endpoint url in browser it will give the WSDL, so use WSDL to generate client.

Class will have @WebService and method will have @WebMethod

@WebMethod	action – value of soap action exclude – true/false operationName - <wsdl:operation
@WebService	name - <wsdl:portType name= portName - <wsdl:port name= serviceName - <wsdl:service name= targetNamespace – namespace for <wsdl:portType endpointInterface – complete name of service endpoint wsdlLocation – location of pre-defined WSDL  If class has @WebService then all public methods are automatically webmethods, no need to explicitly put @WebMethod for each method.
@Oneway	means method has only input but no output
@WebParam	(name – maps to <part name= (header = true means it is a header field
@WebFault	maps to <wsdl:fault
@WebResult	return value of a webmethod
@postConstruct	This method is executed after dependency injection is applied on the class
@preDestroy	This method is executed just before instance is being removed by container
@Resource	Maps webservicecontext resource to a field or method

JAX RPC	JAX WS
Cannot support asynchronous and message oriented web services	Supports asynchronous and message oriented web services
Uses java type binding and only 90% XML	100% XML with JAXB
No annotations and Deployment Descriptor is must	Uses annotations and Deployment Descriptor is optional

### Advantages of webservices:

- JAXWS has API to access the header info
- Stateless so only one instance available for all callers, so thread safe

### Disadvantages of webservices:

- Need to know XML
- All data types to be XML types
- Marshalling and unmarshalling overhead
- Distributed transaction management is not possible. E.g. there are 3 method calls and method2 call is to a webservice. If method 3 fails, then there is no way to tell the webmethod to rollback.

SOAP engines – Apache Axis, JBoss WS

JBoss 5.0 is the best app server for webservices.

REST	SOAP
Expose resources which represent data	Expose operations which represent logic
Uses HTTP (GET/POST/DELETE)	Uses HTTP POST only
Point-to-point tunneling	Distributed messaging
Supports multiple data formats	Supports only XML
Supports stateless communication only	Supports stateless , stateful and asynchronous messaging
Lightweight does not require XML parsing	

Not secure, does not use contract(WSDL)	
-----------------------------------------	--

## **BUSINESS PROCESS MANAGENT**

WBM 6.0

ALBPM 6.0

Modeling modes

- Basic
- Advanced

Synchronize is used to resolve datatype mismatches between the subprocess and the called process.

Model published may not be available in Reviewing space immediately because the job may still be in queue and not yet published.

Depending on size of model, Designer space supports 30 concurrent users and Review space supports 20 concurrent users.

Export from latest version cannot be imported into older version Modeler

Process catalog and Data catalog can be exported only by using Report template

If project is large instead of import do migrate whole workspace

Reports generated into PDF may not open in Adobe Acrobat but use other PDF converters while Print to use other readers to open PDFs

WBM Team Synchronize view may not show latest changes done to artifacts if they are shared to Team Repository from another machine whose clock is different from machine where you are checking.

### **Pallette**

- Start node
- Stop node
- Terminate node
- Single choice decision
- Multiple choice decision
- Local task
- Human task
- Rules task
- Process
- Subprocess
- Annotation
- FOR loop
- WHILE loop
- DO WHILE loop
- IF THEN ELSE decision box
- Merge
- Fork
- Join

### **Parts of the interface in WBM**

- Project Tree view
- Process Editor
- Pallette
- Outline view
- Properties view

### **Parts of a Model**

- Business items(entities)
- Processes
- Resources
- Reports
- Queries
- Business services
- Business service objects

http://www.utorialspoint.com/nodejs/nodejs\_interview\_questions.htm  
Copyright © tut

```
<% for(var i=0; i<5; i++) { %>
 Yunus
<% } %>
```

<h1><%= title %></h1>

<%= 28%> <-- only for variables -->

if you create EJS

then create JS also in routes folder

Dear readers, these **Node.JS Interview Questions** have been designed specially to get you acquainted with the nature of questions you may encounter during your interview for the subject of **Node.JS**. As per my experience good interviewers hardly plan to ask any particular question during your interview, normally questions start with some basic concept of the subject and later they continue based on further discussion and what you answer:

**Tools - Jetbrains IDE IntelliJ IDEA (trial)**  
**Version 12.1.7 Written in Java**  
**we use it for open source projects.....**

What is Node.js?

Node.js is a web application framework built on Google Chrome's JavaScript Engine V8 Engine.

Node.js comes with runtime environment on which a Javascript based script can be interpreted and executed. It is analogous to JVM to Java bytecode. This runtime allows to execute a JavaScript code on any machine outside a browser. Because of this runtime of Node.js, JavaScript is now can be executed on server as well.

**modules = different files**

**THIS KEYWORD**

Node.js also provides a rich library of various javascript modules which eases the development of web application using Node.js to great extents.

Node.js = Runtime Environment + JavaScript Library

What do you mean by Asynchronous API?

```
var First= {
 name: function F()
 {
 console.log("Welcome message");
 console.log(this === First);
 First.name();
 }
}

// 5 eggs - server - chef -
// 5@5sec
function F2()
{
 5 eggs - 5@5 sec
 console.log("Welcome message");
}
```

What are the benefits of using Node.js?

Following are main benefits of using Node.js

**It is very popular because in a web applications --> many requests is coming**

- Aynchronous and Event Driven** All APIs of Node.js library are asynchronous. It essentially means a Node.js based server never waits for a API to return data. Server moves to next API after calling it and a notification mechanism of Events of Node.js helps server to get response from the previous API call.
- Very Fast** Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- Single Threaded but highly Scalable** - Node.js uses a single threaded model with event looping. Event mechanism helps server to respond in a non-blocking ways and makes server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and same program can services much larger number of requests than traditional server like Apache HTTP Server.
- No Buffering** - Node.js applications never buffer any data. These applications simply output the data in chunks.

```
MODULE.JS
var Finalmodule = require("./module2");
Finalmodule.subModule();
```

```
MODULE2.JS
function subModule1()
{
 console.log("This is sub module 1");
}
```

```
METHOD 2:
var Finalmodule = require("./module2");
Finalmodule.subModule1();
console.log(Finalmodule.subModule2);
```

```
function subModule2()
{
 console.log("This is sub module 2");
}
```

```
MODULE2.JS
module.exports = {
 subModule1: function()
 {
 console.log("This is sub module 1");
 },
 subModule2: "This is variable sub module2",
};
```

Is it free to use Node.js?

Yes! Node.js is released under the [MIT license](#) and is free to use.

Is Node a single threaded application?

Yes! Node uses a single threaded model with event looping.

What is REPL in context of Node?

REPL stands for Read Eval Print Loop and it represents a computer environment like a window console or unix/linux shell where a command is entered and system responds with an output. Node.js or Node comes bundled with a REPL environment. It performs the following desired tasks.

- **Read** - Reads user's input, parse the input into JavaScript data-structure and stores in memory.
- **Eval** - Takes and evaluates the data structure
- **Print** - Prints the result
- **Loop** - Loops the above command until user press ctrl-c twice.

Can we evaluate simple expression using Node REPL

Yes.

#### OBJECT FACTORY:

```
MODULE2.JS
module.exports= function(){
 return {
 favMovie: ""
 }
}
```

```
MODULE3.JS:
var Module3 = require("./module2");
```

```
var Module3Object = Module3();
Module3Object.favMovie="Titanic";
console.log("Module3 fav movie is : "+Module3Object.favMovie);
```

```
MODULE4.JS: same as MODULE3.JS
```

What is the difference of using var and not using var in REPL while dealing with variables?

Use variables to store values and print later. if var keyword is not used then value is stored in the variable and printed. Whereas if var keyword is used then value is stored but not printed. You can use both variables later.

What is the use of Underscore variable in REPL?

Use \_ to get the last result.

```
C:\Nodejs_WorkSpace>node
> var x = 10
undefined
> var y = 20
undefined
> x + y
30
> var sum = _
undefined
> console.log(sum)
30
undefined
>
```

What is npm?

npm stands for Node Package Manager. npm provides following two main functionalities:

- Online repositories for node.js packages/modules which are searchable on [search.nodejs.org](https://search.nodejs.org)
- Command line utility to install packages, do version management and dependency management of Node.js packages.

What is global installation of dependencies?

Globally installed packages/dependencies are stored in <user-directory>/npm directory. Such dependencies can be used in CLI CommandLineInterface function of any node.js but can not be imported using require in Node application directly. To install a Node project globally use -g flag.

```
C:\Nodejs_WorkSpace>npm install express -g
```

[In terminal type to connect](#)

What is local installation of dependencies?

[-> npm install connect](#)

By default, npm installs any dependency in the local mode. Here local mode refers to the package installation in node\_modules directory lying in the folder where Node application is present. Locally deployed packages are accessible via require. To install a Node project locally following is the syntax.

```
C:\Nodejs_WorkSpace>npm install express
```

How to check the already installed dependencies which are globally installed using npm?

Use the following command:

```
C:\Nodejs_WorkSpace>npm ls -g
```

What is Package.json?

package.json is present in the root directory of any Node application/module and is used to define the properties of a package.

Name some of the attributes of package.json?

Following are the attributes of Package.json

- **name** - name of the package
- **version** - version of the package
- **description** - description of the package
- **homepage** - homepage of the package
- **author** - author of the package
- **contributors** - name of the contributors to the package
- **dependencies** - list of dependencies. npm automatically installs all the dependencies mentioned here in the node\_module folder of the package.
- **repository** - repository type and url of the package
- **main** - entry point of the package
- **keywords** - keywords

How to uninstall a dependency using npm?

Use following command to uninstall a module.

```
C:\Nodejs_WorkSpace>npm uninstall dependency-name
```

How to update a dependency using npm?

Update package.json and change the version of the dependency which to be updated and run the following command.

```
C:\Nodejs_WorkSpace>npm update
```

What is Callback? **server tells the DB "chef" just call me back once the eggs are ready... till that i will doing some other work. In real time, number of users accessing the DB at a time....**

Callback is an asynchronous equivalent for a function. A callback function is called at the completion of a given task. Node makes heavy use of callbacks. All APIs of Node are written in such a way that they supports

```
function PlaceanOrder(ordernumber)
{
 console.log("Customer order No: "+ordernumber);

 CookingTime(function()
 {
 console.log("Deliver order No: "+ordernumber);
 })
}

setTimeout(callback,5000);

PlaceanOrder(1);
PlaceanOrder(2);
PlaceanOrder(3);
PlaceanOrder(4);
PlaceanOrder(5);
```

callbacks. For example, a function to read a file may start reading file and return the control to execution environment immidiately so that next instruction can be executed. Once file I/O is complete, it will call the callback function while passing the callback function, the content of the file as parameter. So there is no blocking or wait for File I/O. This makes Node.js highly scalable, as it can process high number of request without waiting for any function to return result.

What is a blocking code?

If application has to wait for some I/O operation in order to complete its execution any further then the code responsible for waiting is known as blocking code.

How Node prevents blocking code?

By providing callback function. Callback function gets called whenever corresponding event triggered.

What is Event Loop?

Node js is a single threaded application but it support concurrency via concept of event and callbacks. As every API of Node js are asynchronous and being a single thread, it uses async function calls to maintain the concurrency. Node uses observer pattern. Node thread keeps an event loop and whenever any task get completed, it fires the corresponding event which signals the event listener function to get executed.

What is Event Emmitter?

EventEmitter class lies in **events** module. It is accessibly via following syntax:

**Understanding References to Objects:**

```
//import events module
var events = require('events');
//create an eventEmitter object
var eventEmitter = new events.EventEmitter();
```

```
var name= {
 name: "Yunus",
 age: "24"
}

var name2 = name;
name2.name = "New Yunus";

console.log(name.name);
```

When an EventEmitter instance faces any error, it emits an 'error' event. When new listener is added, 'newListener' event is fired and when a listener is removed, 'removeListener' event is fired.

EventEmitter provides multiple properties like **on** and **emit**. **on** property is used to bind a function with the event and **emit** is used to fire an event.

What is purpose of Buffer class in Node?

Buffer class is a global class and can be accessed in application without importing buffer module. A Buffer is a kind of an array of integers and corresponds to a raw memory allocation outside the V8 heap. A Buffer cannot be resized.

What is Piping in Node?

Piping is a mechanism to connect output of one stream to another stream. It is normally used to get data from one stream and to pass output of that stream to another stream. There is no limit on piping operations. Consider the above example, where we've read test.txt using readerStream and write test1.txt using writerStream. Now we'll use the piping to simplify our operation or reading from one file and writing to another file.

Which module is used for file based operations?

fs module is used for file based operations.

```
var fs = require("fs")
```

Which module is used for buffer based operations?

buffer module is used for buffer based operations.

```
var buffer = require("buffer")
```

Which module is used for web based operations?

http module is used for web based operations.

```
var http = require("http")
```

fs module provides both synchronous as well as asynchronous methods.

true.

What is difference between synchronous and asynchronous method of fs module?

Every method in fs module have synchronous as well as asynchronous form. Asynchronous methods takes a last parameter as completion function callback and first parameter of the callback function is error. It is preferred to use asynchronous method instead of synchronous method as former never block the program execution where the latter one does.

Name some of the flags used in read/write operation on files.

flags for read/write operations are following:

- **r** - Open file for reading. An exception occurs if the file does not exist.
- **r+** - Open file for reading and writing. An exception occurs if the file does not exist.
- **rs** - Open file for reading in synchronous mode. Instructs the operating system to bypass the local file system cache. This is primarily useful for opening files on NFS mounts as it allows you to skip the potentially stale local cache. It has a very real impact on I/O performance so don't use this flag unless you need it. Note that this doesn't turn fs.open into a synchronous blocking call. If that's what you want then you should be using fs.openSync
- **rs+** - Open file for reading and writing, telling the OS to open it synchronously. See notes for 'rs' about using this with caution.
- **w** - Open file for writing. The file is created if it does not exist or truncated if it exists.
- **wx** - Like 'w' but fails if path exists.
- **w+** - Open file for reading and writing. The file is created if it does not exist or truncated if it exists.
- **wx+** - Like 'w+' but fails if path exists.
- **a** - Open file for appending. The file is created if it does not exist.
- **ax** - Like 'a' but fails if path exists.
- **a+** - Open file for reading and appending. The file is created if it does not exist.
- **ax+** - Like 'a+' but fails if path exists.

## What are streams?

Streams are objects that let you read data from a source or write data to a destination in continuous fashion.

How many types of streams are present in Node.

In Node.js, there are four types of streams.

- **Readable** - Stream which is used for read operation.
- **Writable** - Stream which is used for write operation.
- **Duplex** - Stream which can be used for both read and write operation.
- **Transform** - A type of duplex stream where the output is computed based on input.

Name some of the events fired by streams.

Each type of Stream is an **EventEmitter** instance and throws several events at different instances of time. For example, some of the commonly used events are:

- **data** - This event is fired when there is data available to read.
- **end** - This event is fired when there is no more data to read.
- **error** - This event is fired when there is any error receiving or writing data.
- **finish** - This event is fired when all data has been flushed to underlying system

What is Chaining in Node?

Chaining is a mechanism to connect output of one stream to another stream and create a chain of multiple stream operations. It is normally used with piping operations.

How will you open a file using Node?

Following is the syntax of the method to open a file in asynchronous mode:

```
fs.open(path, flags[, mode], callback)
```

## Parameters

Here is the description of the parameters used:

- **path** - This is string having file name including path.
- **flags** - Flag tells the behavior of the file to be opened. All possible values have been mentioned below.
- **mode** - This sets the file mode permission and sticky bits, but only if the file was created. It defaults to 0666, readable and writable.
- **callback** - This is the callback function which gets two arguments err, fd.

How will you read a file using Node?

Following is the syntax of one of the methods to read from a file:

```
fs.read(fd, buffer, offset, length, position, callback)
```

This method will use file descriptor to read the file, if you want to read file using file name directly then you should use another method available.

## Parameters

Here is the description of the parameters used:

- **fd** - This is the file descriptor returned by file fs.open method.
- **buffer** - This is the buffer that the data will be written to.
- **offset** - This is the offset in the buffer to start writing at.
- **length** - This is an integer specifying the number of bytes to read.
- **position** - This is an integer specifying where to begin reading from in the file. If position is null, data will be read from the current file position.
- **callback** - This is the callback function which gets the three arguments, err,bytesRead,buffer.

How will you write a file using Node?

Following is the syntax of one of the methods to write into a file:

```
fs.writeFile(filename, data[, options], callback)
```

This method will over-write the file if file already exists. If you want to write into an existing file then you should use another method available.

## Parameters

Here is the description of the parameters used:

- **path** - This is string having file name including path.
- **data** - This is the String or Buffer to be written into the file.
- **options** - The third parameter is an object which will hold {encoding, mode, flag}. By default encoding is utf8, mode is octal value 0666 and flag is 'w'
- **callback** - This is the callback function which gets a single parameter err and used to return error in case of any writing error.

How will you close a file using Node?

Following is the syntax of one of the methods to close an opened file:

```
fs.close(fd, callback)
```

## Parameters

Here is the description of the parameters used:

- **fd** - This is the file descriptor returned by file fs.open method.
- **callback** - This is the callback function which gets no arguments other than a possible exception are given to the completion callback.

How will you get information about a file using Node?

Following is the syntax of the method to get the information about a file:

```
fs.stat(path, callback)
```

## Parameters

Here is the description of the parameters used:

- **path** - This is string having file name including path.
- **callback** - This is the callback function which gets two arguments err,stats where **stats** is an object of fs.Stats type which is printed below in the example.

How will you truncate a file using Node?

Following is the syntax of the method to truncate an opened file:

```
fs.ftruncate(fd, len, callback)
```

## Parameters

Here is the description of the parameters used:

- **fd** - This is the file descriptor returned by file fs.open method.
- **len** - This is the length of the file after which file will be truncated.
- **callback** - This is the callback function which gets no arguments other than a possible exception are given to the completion callback.

How will you delete a file using Node?

Following is the syntax of the method to delete a file:

```
fs.unlink(path, callback)
```

## Parameters

Here is the description of the parameters used:

- **path** - This is the file name including path.
- **callback** - This is the callback function which gets no arguments other than a possible exception are given to the completion callback.

How will you create a directory?

Following is the syntax of the method to create a directory:

```
fs.mkdir(path[, mode], callback)
```

## Parameters

Here is the description of the parameters used:

- **path** - This is the directory name including path.
- **mode** - This is the directory permission to be set. Defaults to 0777.
- **callback** - This is the callback function which gets no arguments other than a possible exception are given to the completion callback.

How will you delete a directory?

Following is the syntax of the method to remove a directory:

```
fs.rmdir(path, callback)
```

## Parameters

Here is the description of the parameters used:

- **path** - This is the directory name including path.
- **callback** - This is the callback function which gets no arguments other than a possible exception are given to the completion callback.

How will you read a directory?

Following is the syntax of the method to read a directory:

```
fs.readdir(path, callback)
```

## Parameters

Here is the description of the parameters used:

- **path** - This is the directory name including path.
- **callback** - This is the callback function which gets two arguments err,files where files is an array of the names of the files in the directory excluding '.' and '..'.

What is the purpose of \_\_filename variable?

The \_\_filename represents the filename of the code being executed. This is the resolved absolute path of this code file. For a main program this is not necessarily the same filename used in the command line. The value inside a module is the path to that module file.

What is the purpose of \_\_dirname variable?

The \_\_dirname represents the name of the directory that the currently executing script resides in.

What is the purpose of setTimeout function?

The setTimeoutcb,ms global function is used to run callback cb after at least ms milliseconds. The actual delay depends on external factors like OS timer granularity and system load. A timer cannot span more than 24.8 days.

This function returns an opaque value that represents the timer which can be used to clear the timer.

What is the purpose of clearTimeout function?

The clearTimeout global function is used to stop a timer that was previously created with setTimeout. Here t is the timer returned by setTimeout function.

What is the purpose of setInterval function?

The setIntervalcb,ms global function is used to run callback cb repeatedly after at least ms milliseconds. The actual delay depends on external factors like OS timer granularity and system load. A timer cannot span more than 24.8 days.

This function returns an opaque value that represents the timer which can be used to clear the timer using the function clearInterval.

What is the purpose of console object?

console object is used to Used to print information on stdout and stderr.

What is the purpose of process object?

process object is used to get information on current process. Provides multiple events related to process activities.

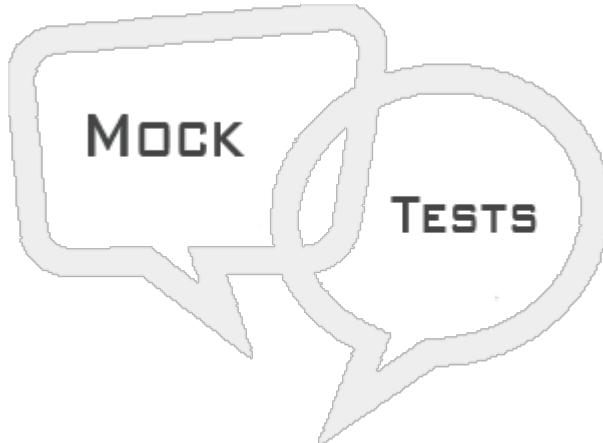
## What is Next ?

Further you can go through your past assignments you have done with the subject and make sure you are able to speak confidently on them. If you are fresher then interviewer does not expect you will answer very complex questions, rather you have to make your basics concepts very strong.

Second it really doesn't matter much if you could not answer few questions but it matters that whatever you answered, you must have answered with confidence. So just feel confident during your interview. We at tutorialspoint wish you best luck to have a good interviewer and all the very best for your future endeavor. Cheers :-)

# NODE.JS MOCK TEST

This section presents you various set of Mock Tests related to **Node.js Framework**. You can download these sample mock tests at your local machine and solve offline at your convenience. Every mock test is supplied with a mock test key to let you verify the final score and grade yourself.



## NODE.JS MOCK TEST I

### **Q 1 - Which of the following is true about Node.JS?**

- A - Node.js is a JavaScript based framework/platform built on Google Chrome's JavaScript V8 Engine.
- B - Node.JS is used to develop I/O intensive web applications like video streaming sites, single page applications and other web application.
- C - Node.js is open source and is completely free to use.
- D - All of the above.

### **Q 2 - What is Node.JS?**

- A - Node.js is a web server.
- B - Node.js is a JavaScript based framework/platform built on Google Chrome's JavaScript V8 Engine.
- C - Node.js is a java based framework.
- D - None of the above.

### **Q 3 - All APIs of Node.JS are.**

- A - Asynchronous
- B - Synchronous
- C - Both of the above.
- D - None of the above.

### **Q 4 - Why code written in Node.JS is pretty fast although being written in JavaScript?**

- A - Node.JS internally converts JavaScript code to Java based code and then execute the same.
- B - Node.JS internally converts JavaScript code to C based code and then execute the same.
- C - Being built on Google Chrome's V8 JavaScript Engine.
- D - None of the above.

**Q 5 - How Node based web servers are different from traditional web servers?**

- A - Node based server process request much faster than traditional server.
- B - Node based server uses a single threaded model and can services much larger number of requests than traditional server like Apache HTTP Server.
- C - There is no much difference between the two.
- D - None of the above.

**Q 6 In which of the following areas, Node.js is perfect to use?**

- A - I/O bound Applications
- B - Data Streaming Applications
- C - Data Intensive Realtime Applications (DIRT)
- D - All of the above.

**Q 7 In which of the following areas, Node.js is not advised to be used?**

- A - Single Page Applications
- B - JSON APIs based Applications
- C - CPU intensive applications
- D - Data Intensive Realtime Applications (DIRT)

**Q 8 Which of the following statement is valid to use a Node module http in a Node based application?**

- C - package http;
- D - import http;

**Q 9 REPL stands for.**

- A - Research Eval Program Learn
- B - Read Eval Print Loop
- C - Read Earn Point Learn
- D - Read Eval Point Loop

**Q 10 Which of following command starts a REPL session?**

- A - \$ node
- B - \$ node start
- C - \$ node repl
- D - \$ node console

**Q 11 - What is use of Underscore Variable in REPL session?**

- A - to get the last command used.
- B - to get the last result.
- C - to store the result.
- D - None of the above.

**Q 12 -What npm stands for?**

- A - Node Package Manager
- B - Node Project Manager
- C - New Project Manager
- D - New Package Manager

**Q 13 - Which of the following command will show version of Node?**

- A - \$ npm --version
- B - \$ node --version
- C - \$ npm getVersion
- D - \$ node getVersion

**Explanation**

**Executing \$ node --version command will show the version of Node instance.**

**Q 14 - Which of the following command will show version of npm?**

- A - \$ npm --version
- B - \$ node --version
- C - \$ npm getVersion
- D - \$ node getVersion

**Q 15 - By default, npm installs any dependency in the local mode.**

- A - true
- B - false

**Q 16 - By default, npm installs any dependency in the global mode.**

A - true

B - false

**Q 17 - Which of the following command will show all the modules installed globally?**

A - \$ npm ls -g

B - \$ npm ls

C - \$ node ls -g

D - \$ node ls

**Q 18 - Which of the following command will show all the modules installed locally.**

A - \$ npm ls -g

B - \$ npm ls

C - \$ node ls -g

D - \$ node ls

**Q 19 - Which of the following is true about package.json?**

A - package.json is present in the root directory of any Node application/module.

B - package.json is used to define the properties of a package.

C - package.json can be used to update dependencies of a Node application.

D - All of the above.

**Q 20 - What is Callback?**

A - Callback is an asynchronous equivalent for a function.

B - Callback is a technique in which a method call back the caller method.

C - Both of the above.

D - None of the above.

**Q 21 - Node js is a single threaded application but supports concurrency.**

A - true

B - false

**Q 22 - Which of the following is true with respect to Node.**

A - Every API of Node.js are asynchronous.

B - Node being a single thread, and uses async function calls to maintain the concurrency.

C - Node thread keeps an event loop and whenever any task gets completed, it fires the corresponding event which signals the event listener function to get executed.

D - All of the above.

**Q 23 - Which of the following provides in-built events.**

A - events

B - callback

C - throw

D - handler

**Q 24 - Which of the following is true about EventEmitter.on property?**

A - on property is used to fire event.

B - on property is used to bind a function with the event.

C - on property is used to locate an event handler.

D - None of the above.

**Q 25 - Which of the following is true about EventEmitter.emit property?**

A - emit property is used to locate an event handler.

B - emit property is used to bind a function with the event.

C - emit property is used to fire an event.

D - None of the above.

## ANSWER SHEET

---

**Question Number      Answer Key**

1                    D

2                    B

3                    A

4                    C

5                    B

6                    D

7                    C

8                    A

9                    B

10	A
11	B
12	A
13	B
14	A
15	A
16	B
17	A
18	B
19	D
20	A
21	A
22	B
23	A
24	B
25	C

## N O D E . - I N T R O D U C T I O N

### What is Node.js?

Node.js is a web application framework built on Google Chrome's JavaScript Engine *V8Engine*. Its latest version is v0.10.36. Definition of Node.js as put by its [official documentation](#) is as follows:

*Node.js® is a platform built on [Chrome's JavaScript runtime](#) for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.*

Node.js comes with runtime environment on which a Javascript based script can be interpreted and executed *It is analogous to JVM to Java bytecode*. This runtime allows to execute a JavaScript code on any machine outside a browser. Because of this runtime of Node.js, JavaScript is now can be executed on server as well.

Node.js also provides a rich library of various javascript modules which eases the development of web application using Node.js to great extents.

Node.js = Runtime Environment + JavaScript Library

### Features of Node.js

- **Aynchronous and Event Driven** All APIs of Node.js library are asynchronous that is non-blocking. It essentially means a Node.js based server never waits for a API to return data. Server moves to next API after calling it and a notification mechanism of Events of Node.js helps server to get response from the previous API call.
- **Very Fast** Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- **Single Threaded but highly Scalable** - Node.js uses a single threaded model with event looping. Event mechanism helps server to respond in a non-blocking ways and makes server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and same program can service much larger number of requests than traditional server like Apache HTTP Server.
- **No Buffering** - Node.js applications never buffer any data. These applications simply output the data in chunks.
- **License** - Node.js is released under the [MIT license](#).

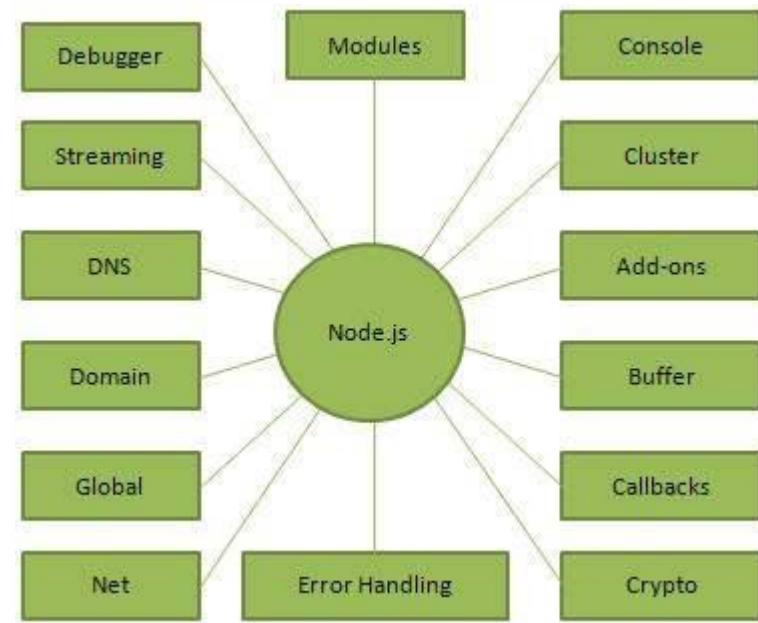
### Who Uses Node.js?

Following is the link on github wiki containing an exhaustive list of projects, application and companies which are using Node.js. This list include eBay, General Electric, GoDaddy, Microsoft, PayPal, Uber, Wikispaces, Yahoo!, Yammer and the list continues.

[Projects, Applications, and Companies Using Node](#)

## Concepts

The following diagram depicts some important parts of Node.js which we will discuss in detail in the subsequent chapters.



## Where to Use Node.js?

Following are the areas where Node.js is proving itself a perfect technology partner.

- I/O bound Applications
- Data Streaming Applications
- Data Intensive Realtime Applications *DIRT*
- JSON APIs based Applications
- Single Page Applications

## Where Not to Use Node.js?

It is not advisable to use Node.js for CPU intensive applications.

## N O D E . - E N V I R O N M E N T   S E T U P

---

### Try it Option Online

You really do not need to set up your own environment to start learning Node.js. Reason is very simple, we already have set up Node.js environment online, so that you can compile and execute all the available examples online at the same time when you are doing your theory work. This gives you confidence in what you are reading and to check the result with different options. Feel free to modify any example and execute it online.

Try following example using **Try it** option available at the top right corner of the below sample code box:

```
console.log("Hello World!");
```

For most of the examples given in this tutorial, you will find **Try it** option, so just make use of it and enjoy your learning.

## Local Environment Setup

If you are still willing to set up your environment for Node.js, you need the following two softwares available on your computer, *a* Text Editor and *b* The Node.js binary installables.

### Text Editor

This will be used to type your program. Examples of few editors include Windows Notepad, OS Edit command, Brief, Epsilon, EMACS, and vim or vi.

Name and version of text editor can vary on different operating systems. For example, Notepad will be used on Windows, and vim or vi can be used on windows as well as Linux or UNIX.

The files you create with your editor are called source files and contain program source code. The source files for Node.js programs are typically named with the extension ".js".

Before starting your programming, make sure you have one text editor in place and you have enough experience to write a computer program, save it in a file, compile it and finally execute it.

### The Node.js Runtime

The source code written in source file is simply javascript. The Node.js interprter will be used to interpret and execute your javascript code.

Node.js distribution comes as a binary installable for SunOS , Linux, Mac OS X, and Windows operating systems with the 32-bit 386 and 64-bit *amd64* x86 processor architectures.

Following section guides you on how to install Node.js binary distribution on various OS.

### Download Node.js archive

Download latest version of Node.js installable archive file from [Node.js Downloads](#). At the time of writing this tutorial, I downloaded *node-v0.12.0-x64.msi* and copied it into C:\>nodejs folder.

OS	Archive name
Windows	node-v0.12.0-x64.msi

Linux	node-v0.12.0-linux-x86.tar.gz
Mac	node-v0.12.0-darwin-x86.tar.gz
SunOS	node-v0.12.0-sunos-x86.tar.gz

## Installation on UNIX/Linux/Mac OS X, and SunOS

Extract the download archive into /usr/local, creating a Node.js tree in /usr/local/nodejs. For example:

```
tar -C /usr/local -xzf node-v0.12.0-linux-x86.tar.gz
```

Add /usr/local/nodejs to the PATH environment variable.

OS	Output
Linux	export PATH=\$PATH:/usr/local/nodejs
Mac	export PATH=\$PATH:/usr/local/nodejs
FreeBSD	export PATH=\$PATH:/usr/local/nodejs

## Installation on Windows

Use the MSI file and follow the prompts to install the Node.js. By default, the installer uses the Node.js distribution in C:\Program Files\nodejs. The installer should set the C:\Program Files\nodejs directory in window's PATH environment variable. Restart any open command prompts for the change to take effect.

## Verify installation: Executing a File

Create a js file named test.js in **C:\>Nodejs\_WorkSpace**.

*File: test.js*

```
console.log("Hello World")
```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

Verify the Output

```
Hello, World!
```

N O D E . - H I S R S T   A P P L I C A T I O N

Before creating actual Hello World ! application using Node.js, let us see the parts of a Node.js application. A Node.js application consists of following three important parts:

- **import required module:** use require directive to load a javascript module
- **create server:** A server which will listen to client's request similar to Apache HTTP Server.
- **read request and return response:** server created in earlier step will read HTTP request made by client which can be a browser or console and return the response.

## Creating Node.js Application

Step 1: import required module

use require directive to load http module.

```
var http = require("http")
```

Step 2: create an HTTP server using http.createServer method. Pass it a function with parameters request and response. Write the sample implementation to always return "Hello World". Pass a port 8081 to listen method.

```
http.createServer(function (request, response) {
 // HTTP Status: 200 : OK
 // Content Type: text/plain
 response.writeHead(200, {'Content-Type': 'text/plain'});
 // send the response body as "Hello World"
 response.end('Hello World\n');
}).listen(8081);
// console will print the message
console.log('Server running at http://127.0.0.1:8081/');
```

```
var http = require('http');

function onRequest(request,response)
{
 // accessing the information
 //
 console.log("User made a request...."+request.url);
 response.writeHead(200,{Content-Type: "text/plain"});
 // this can be file also
 response.write("Here is some data....");
 // write in web page
 response.end();
}
http.createServer(onRequest).listen(8888);
//listening through port number.
console.log("Server is running");
// parameter is the code you need to run and retrieve.....
```

Step 3: Create a js file named test.js in **C:\>Nodejs\_WorkSpace**.

```
var http = require("http")
http.createServer(function (request, response) {
 response.writeHead(200, {'Content-Type': 'text/plain'});
 response.end('Hello World\n');
}).listen(8081);
console.log('Server running at http://127.0.0.1:8081/');
```

**OUTPUT:**  
**Server is running**  
**User made a request..../**  
**User made a request..../favicon.ico**

**SIMPLE SERVER WITH HTML FILE**

```
// hosting a website...
var http = require('http');
var fs = require('fs');

// built a 404 response for error handling.....
function send404Response(response)
{
 response.writeHead(404, {"content-type": "text/plain"});
 response.write("ERROR: 404 PAGE NOT FOUND.....");
 response.end();
}

function onRequest(request,response)
{
 if(request.method == 'GET' && request.url == '/')
 //connecting to the homepage
 {
 response.writeHead(200, {"content-type": "text/plain"});
 fs.createReadStream('E:\IdeaProjects\FirstApplication\SimpleServerIndex.html').pipe(response);
 // reading the stream of the html page....
 }
 else
 {
 send404Response(response);
 }
}

http.createServer(onRequest).listen(8888);
console.log("simple server is running....");
```

Now run the test.js to see the result:

C:\Nodejs\_WorkSpace>node test.js

Verify the Output. Server has started

Server running at http://127.0.0.1:8081/

**Make a request to Node.js server**

Open http://127.0.0.1:8081/ in any browser and see the below result

**N O D.E J - R E P L**

REPL stands for Read Eval Print Loop and it represents a computer environment like a window console or unix/linux shell where a command is entered and system responds with an output. Node.js or Node comes bundled with a REPL environment. It performs the following desired tasks.

- **Read** - Reads user's input, parse the input into JavaScript data-structure and stores in memory.
- **Eval** - Takes and evaluates the data structure
- **Print** - Prints the result
- **Loop** - Loops the above command until user press ctrl-c twice.

REPL feature of Node is very useful in experimenting with Node.js codes and to debug JavaScript codes.

## Features

REPL can be started by simply running node on shell/console without any argument.

```
C:\Nodejs_WorkSpace> node
```

You will see the REPL Command prompt:

```
C:\Nodejs_WorkSpace> node
>
```

## Simple Expression

Let's try simple mathematics at REPL command prompt:

```
C:\Nodejs_WorkSpace>node
> 1 + 3
4
> 1 + (2 * 3) - 4
3
>
```

## Use variables

Use variables to store values and print later. if var keyword is not used then value is stored in the variable and printed. Whereas if var keyword is used then value is stored but not printed. You can use both variables later. Print anything using console.log

```
C:\Nodejs_WorkSpace> node
> x = 10
10
> var y = 10
undefined
> x + y
20
> console.log("Hello World")
Hello Workd
undefined
```

## Multiline Expression

Node REPL supports multiline expression similar to JavaScript. See the following do-while loop in action:

```
C:\Nodejs_WorkSpace> node
> var x = 0
undefined
> do {
... x++;
... console.log("x: " + x);
... } while (x < 5);
x: 1
x: 2
x: 3
x: 4
```

```
x: 5
undefined
>
```

... comes automatically when you press enters after opening bracket. Node automatically checks the continuity of expressions.

## Underscore variable

Use `_` to get the last result.

```
C:\Nodejs_WorkSpace>node
> var x = 10
undefined
> var y = 20
undefined
> x + y
30
> var sum = _
undefined
> console.log(sum)
30
undefined
>
```

## REPL Commands

- **ctrl + c** - terminate the current command.
- **ctrl + c twice** - terminate the Node REPL.
- **ctrl + d** - terminate the Node REPL.
- **Up/Down Keys** - see command history and modify previous commands.
- **tab Keys** - list of current commands.
- **.help** - list of all commands.
- **.break** - exit from multiline expression.
- **.clear** - exit from multiline expression
- **.save** - save current Node REPL session to a file.
- **.load** - load file content in current Node REPL session.

```
C:\Nodejs_WorkSpace>node
> var x = 10
undefined
> var y = 20
undefined
> x + y
30
> var sum = _
undefined
> console.log(sum)
30
undefined
> .save test.js
Session saved to:test.js
> .load test.js
> var x = 10
undefined
> var y = 20
undefined
> x + y
30
```

```
> var sum =
undefined
> console.log(sum)
30
undefined
>
```

## N O D E . - N I S M

npm stands for Node Package Manager. npm provides following two main functionalities:

- Online repositories for node.js packages/modules which are searchable on [search.nodejs.org](https://search.nodejs.org)
- Command line utility to install packages, do version management and dependency management of Node.js packages.

npm comes bundled with Node.js installables after v0.6.3 version. To verify the same, open console and type following command and see the result:

```
C:\Nodejs_WorkSpace>npm --version
2.5.1
```

### Global vs Local installation

By default, npm installs any dependency in the local mode. Here local mode refers to the package installation in `node_modules` directory lying in the folder where Node application is present. Locally deployed packages are accessible via require.

Globally installed packages/dependencies are stored in `<user-directory>/npm` directory. Such dependencies can be used in `CLI CommandLineInterface` function of any node.js but can not be imported using require in Node application directly.

Let's install express, a popular web framework using local installation.

C:\Program Files\nodejs\node\_modules>npm install express -g

```
C:\Nodejs_WorkSpace>npm install express
express@4.11.2 node_modules\express
|-- merge-descriptors@0.0.2
|-- utils-merge@1.0.0
|-- methods@1.1.1
|-- escape-html@1.0.1
|-- fresh@0.2.4
|-- cookie@0.1.2
|-- range-parser@1.0.2
|-- media-typer@0.3.0
|-- cookie-signature@1.0.5
|-- vary@1.0.0
|-- finalhandler@0.3.3
|-- parseurl@1.3.0
|-- serve-static@1.8.1
|-- content-disposition@0.5.0
|-- path-to-regexp@0.1.3
|-- depd@1.0.0
|-- qs@2.3.3
|-- debug@2.1.1 (ms@0.6.2)
|-- send@0.11.1 (destroy@1.0.3, ms@0.7.0, mime@1.2.11)
|-- on-finished@2.2.0 (ee-first@1.1.0)
|-- type-is@1.5.7 (mime-types@2.0.9)
|-- accepts@1.2.3 (negotiator@0.5.0, mime-types@2.0.9)
|-- etag@1.5.1 (crc@3.2.1)
```

push express modules into npm folder node modules

```
|-- proxy-addr@1.0.6 (forwarded@0.1.0, ipaddr.js@0.1.8)
```

Once npm completes the download, you can verify by looking at the content of C:\Nodejs\_WorkSpace\node\_modules. Or type the following command:

```
C:\Nodejs_WorkSpace>npm ls
C:\Nodejs_WorkSpace
|-- express@4.11.2
| |-- accepts@1.2.3
| | |-- mime-types@2.0.9
| | | |-- mime-db@1.7.0
| | | |-- negotiator@0.5.0
| | |-- content-disposition@0.5.0
| | |-- cookie@0.1.2
| | |-- cookie-signature@1.0.5
| | |-- debug@2.1.1
| | | |-- ms@0.6.2
| | |-- depd@1.0.0
| | |-- escape-html@1.0.1
| | |-- etag@1.5.1
| | | |-- crc@3.2.1
| | |-- finalhandler@0.3.3
| | |-- fresh@0.2.4
| | |-- media-typer@0.3.0
| | |-- merge-descriptors@0.0.2
| | |-- methods@1.1.1
| | |-- on-finished@2.2.0
| | | |-- ee-first@1.1.0
| | |-- parseurl@1.3.0
| | |-- path-to-regexp@0.1.3
| | |-- proxy-addr@1.0.6
| | | |-- forwarded@0.1.0
| | | |-- ipaddr.js@0.1.8
| | |-- qs@2.3.3
| | |-- range-parser@1.0.2
| | |-- send@0.11.1
| | | |-- destroy@1.0.3
| | | |-- mime@1.2.11
| | | |-- ms@0.7.0
| | |-- serve-static@1.8.1
| | |-- type-is@1.5.7
| | | |-- mime-types@2.0.9
| | | | |-- mime-db@1.7.0
| | |-- utils-merge@1.0.0
| | |-- vary@1.0.0
```

Now Let's try installing express, a popular web framework using global installation.

```
C:\Nodejs_WorkSpace>npm install express -g
```

Once npm completes the download, you can verify by looking at the content of <user-directory>/npm/node\_modules. Or type the following command:

```
C:\Nodejs_WorkSpace>npm ls -g
```

## Installing a module

Installation of any module is as simple as typing the following command.

```
C:\Nodejs_WorkSpace>npm install express
```

Now you can use it in your js file as following:

```
var express = require('express');
```

## Using package.json

package.json is present in the root directory of any Node application/module and is used to define the properties of a package. Let's open package.json of express package present in C:\Nodejs\_Workspace\node\_modules\express\

```
{
 "name": "express",
 "description": "Fast, unopinionated, minimalist web framework",
 "version": "4.11.2",
 "author": {
 "name": "TJ Holowaychuk",
 "email": "tj@vision-media.ca"
 },
 "contributors": [
 {
 "name": "Aaron Heckmann",
 "email": "aaron.heckmann+github@gmail.com"
 },
 {
 "name": "Ciaran Jessup",
 "email": "ciaranj@gmail.com"
 },
 {
 "name": "Douglas Christopher Wilson",
 "email": "doug@somethingdoug.com"
 },
 {
 "name": "Guillermo Rauch",
 "email": "rauchg@gmail.com"
 },
 {
 "name": "Jonathan Ong",
 "email": "me@jongleberry.com"
 },
 {
 "name": "Roman Shtylman",
 "email": "romanshtylman@gmail.com"
 }
]
}
```

```
"email": "shtylman+expressjs@gmail.com"
},
{
 "name": "Young Jae Sim",
 "email": "hanul@hanul.me"
}
],
"license": "MIT",
"repository": {
 "type": "git",
 "url": "https://github.com/strongloop/express"
},
"homepage": "http://expressjs.com/",
"keywords": [
 "express",
 "framework",
 "sinatra",
 "web",
 "rest",
 "restful",
 "router",
 "app",
 "api"
],
"dependencies": {
 "accepts": "~1.2.3",
 "content-disposition": "0.5.0",
 "cookie-signature": "1.0.5",
 "debug": "~2.1.1",
 "depd": "~1.0.0",
 "escape-html": "1.0.1",
 "etag": "~1.5.1",
 "finalhandler": "0.3.3",
 "fresh": "0.2.4",
 "media-type": "0.3.0",
 "methods": "~1.1.1",
 "on-finished": "~2.2.0",
 "parseurl": "~1.3.0",
 "path-to-regexp": "0.1.3",
 "proxy-addr": "~1.0.6",
 "qs": "2.3.3",
 "raw-body": "0.2.5",
 "safe-buffer": "2.1.2",
 "serve-static": "2.1.1",
 "type-is": "1.6.1",
 "util-deprecate": "1.0.2"
}
```

```
"range-parser": "~1.0.2",
"send": "0.11.1",
"serve-static": "~1.8.1",
"type-is": "~1.5.6",
"vary": "~1.0.0",
"cookie": "0.1.2",
"merge-descriptors": "0.0.2",
"utils-merge": "1.0.0"
},
"devDependencies": {
 "after": "0.8.1",
 "ejs": "2.1.4",
 "istanbul": "0.3.5",
 "marked": "0.3.3",
 "mocha": "~2.1.0",
 "should": "~4.6.2",
 "supertest": "~0.15.0",
 "hjs": "~0.0.6",
 "body-parser": "~1.11.0",
 "connect-redis": "~2.2.0",
 "cookie-parser": "~1.3.3",
 "express-session": "~1.10.2",
 "jade": "~1.9.1",
 "method-override": "~2.3.1",
 "morgan": "~1.5.1",
 "multiparty": "~4.1.1",
 "vhost": "~3.0.0"
},
"engines": {
 "node": ">= 0.10.0"
},
"files": [
 "LICENSE",
 "History.md",
 "Readme.md",
 "index.js",
 "lib/"
],
"scripts": {
 "test": "mocha --require test/support/env --reporter spec --bail --check-leaks test/test/acceptance/"
}
```

```
"test-cov": "istanbul cover node_modules/mocha/bin/_mocha -- --require test/support/env --reporter dot --check-leaks test/ test/acceptance/",
"test-tap": "mocha --require test/support/env --reporter tap --check-leaks test/ test/acceptance/",
"test-travis": "istanbul cover node_modules/mocha/bin/_mocha --report lcovonly -- --require test/support/env --reporter spec --check-leaks test/ test/acceptance/"
,
"gitHead": "63ab25579bda70b4927a179b580a9c580b6c7ada",
"bugs": {
 "url": "https://github.com/strongloop/express/issues"
,
 "_id": "express@4.11.2",
 "_shasum": "8df3d5a9ac848585f00a0777601823faecd3b148",
 "_from": "express@*",
 "_npmVersion": "1.4.28",
 "_npmUser": {
 "name": "dougwilson",
 "email": "doug@somethingdoug.com"
,
 "maintainers": [
 {
 "name": "tjholowaychuk",
 "email": "tj@vision-media.ca"
,
 {
 "name": "jongleberry",
 "email": "jonathanrichardong@gmail.com"
,
 {
 "name": "shtylman",
 "email": "shtylman@gmail.com"
,
 {
 "name": "dougwilson",
 "email": "doug@somethingdoug.com"
,
 {
 "name": "aredridel",
 "email": "aredridel@nbtsc.org"
,
 {
 "name": "strongloop",
 "email": "strongloop@strongloop.com"
 }
]
}
```

```

 "email": "callback@strongloop.com"
},
{
 "name": "rfeng",
 "email": "enjoyjava@gmail.com"
}
],
"dist": {
 "shasum": "8df3d5a9ac848585f00a0777601823faecd3b148",
 "tarball": "http://registry.npmjs.org/express/-/express-4.11.2.tgz"
},
"directories": {},
"_resolved": "https://registry.npmjs.org/express/-/express-4.11.2.tgz",
"readme": "ERROR: No README data found!"
}

```

## Attributes of Package.json

- **name** - name of the package
- **version** - version of the package
- **description** - description of the package
- **homepage** - homepage of the package
- **author** - author of the package
- **contributors** - name of the contributors to the package
- **dependencies** - list of dependencies. npm automatically installs all the dependencies mentioned here in the node\_module folder of the package.
- **repository** - repository type and url of the package
- **main** - entry point of the package
- **keywords** - keywords

## Uninstalling a module

Use following command to uninstall a module.

```
C:\Nodejs_WorkSpace>npm uninstall express
```

Once npm uninstall the package, you can verify by looking at the content of <user-directory>/npm/node\_modules. Or type the following command:

```
C:\Nodejs_WorkSpace>npm ls
```

## Updating a module

Update package.json and change the version of the dependency which to be updated and run the following command.

```
C:\Nodejs_WorkSpace>npm update
```

## Search a module

Search package name using npm.

```
C:\Nodejs_WorkSpace>npm search express
```

## Create a module

Creation of module requires package.json to be generated. Let's generate package.json using npm.

```
C:\Nodejs_WorkSpace>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sane defaults.

See 'npm help json' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg> --save' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (Nodejs_WorkSpace)
```

Once package.json is generated. Use following command to register yourself with npm repository site using a valid email address.

```
C:\Nodejs_WorkSpace>npm adduser
```

Now its time to publish your module:

```
C:\Nodejs_WorkSpace>npm publish
```

# N O D E . - C A L L B A C K S      C O N C E P T

---

## What is Callback?

Callback is an asynchronous equivalent for a function. A callback function is called at the completion of a given task. Node makes heavy use of callbacks. All APIs of Node are written in such a way that they support callbacks. For example, a function to read a file may start reading file and return the control to execution environment immediately so that next instruction can be executed. Once file I/O is complete, it will call the callback function while passing the callback function, the content of the file as parameter. So there is no blocking or wait for File I/O. This makes Node.js highly scalable, as it can process high number of requests without waiting for any function to return result.

## Blocking Code Example

Create a txt file named test.txt in **C:\>Nodejs\_WorkSpace**

Create a js file named test.js in **C:\>Nodejs\_WorkSpace**

```
var fs = require("fs");
var data = fs.readFileSync('test.txt');
console.log(data.toString());
console.log("Program Ended");
```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

Verify the Output

```
TutorialsPoint.Com
Program Ended
```

## Non-Blocking Code Example

Create a txt file named test.txt in **C:\>Nodejs\_WorkSpace**

```
TutorialsPoint.Com
```

Update test.js in **C:\>Nodejs\_WorkSpace**

```
var fs = require("fs");

fs.readFile('test.txt', function (err, data) {
 if (err) return console.error(err);
 console.log(data.toString());
});

console.log("Program Ended");
```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

Verify the Output

```
Program Ended
TutorialsPoint.Com
```

## Event Loop Overview

Node js is a single threaded application but it support concurrency via concept of event and callbacks. As every API of Node js are asynchronous and being a single thread, it uses async function calls to maintain the concurrency. Node uses observer pattern. Node thread keeps an event loop and whenever any task get completed, it fires the corresponding event which signals the event listener function to get executed.

# Event Driven Programming

Node.js uses Events heavily and it is also one of the reason why Node.js is pretty fast compared to other similar technologies. As soon as Node starts its server, it simply initiates its variables, declares functions and then simply waits for event to occur.

While Events seems similar to what callbacks are. The difference lies in the fact that callback functions are called when an asynchronous function returns its result where event handling works on the observer pattern. The functions which listens to events acts as observer. Whenever an event got fired, its listener function starts executing. Node.js has multiple in-built event. The primary actor is EventEmitter which can be imported using following syntax

```
//import events module
var events = require('events');
//create an eventEmitter object
var eventEmitter = new events.EventEmitter();
```

## Example

Create a js file named test.js in **C:\>Nodejs\_WorkSpace**.

*File: test.js*

```
//import events module
var events = require('events');
//create an eventEmitter object
var eventEmitter = new events.EventEmitter();

//create a function connected which is to be executed
//when 'connection' event occurs
var connected = function connected() {
 console.log('connection successful.');

 // fire the data_received event
 eventEmitter.emit('data_received.');
}

// bind the connection event with the connected function
eventEmitter.on('connection', connected);

// bind the data_received event with the anonymous function
eventEmitter.on('data_received', function() {
 console.log('data received successfully.');
});
```

```
// fire the connection event
eventEmitter.emit('connection');

console.log("Program Ended.");
```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

Verify the Output. Server has started

```
connection succesful.
data received successfully.
Program Ended.
```

## How Node Applications Work?

In Node Application, any async function accepts a callback as a last parameter and the callback function accepts error as a first parameter. Let's revisit the previous example again.

```
var fs = require("fs");

fs.readFile('test.txt', function (err, data) {
 if (err) {
 console.log(err.stack);
 return;
 }
 console.log(data.toString());
});
console.log("Program Ended");
```

Here fs.readFile is a async function whose purpose is to read a file. If an error occur during read of file, then err object will contain the corresponding error else data will contain the contents of the file. readFile passes err and data to callback function after file read operation is complete.

## N O D E . - E V E N T E M I T T E R

---

EventEmitter class lies in **events** module. It is accessibly via following syntax:

```
//import events module
var events = require('events');
//create an eventEmitter object
var eventEmitter = new events.EventEmitter();
```

When an EventEmitter instance faces any error, it emits an 'error' event. When new listener is added, 'newListener' event is fired and when a listener is removed, 'removeListener' event is fired.

EventEmitter provides multiple properties like **on** and **emit**. **on** property is used to bind a function with the event and **emit** is used to fire an event.

## Methods

Sr. No.	method	Description
1	<b>addListener</b> <i>event,listener</i>	Adds a listener to the end of the listeners array for the specified event. No checks are made to see if the listener has already been added. Multiple calls passing the same combination of event and listener will result in the listener being added multiple times. Returns emitter, so calls can be chained.
2	<b>on</b> <i>event,listener</i>	Adds a listener to the end of the listeners array for the specified event. No checks are made to see if the listener has already been added. Multiple calls passing the same combination of event and listener will result in the listener being added multiple times. Returns emitter, so calls can be chained.
3	<b>once</b> <i>event,listener</i>	Adds a one time listener for the event. This listener is invoked only the next time the event is fired, after which it is removed. Returns emitter, so calls can be chained.
4	<b>removeListener</b> <i>event,listener</i>	Remove a listener from the listener array for the specified event. Caution: changes array indices in the listener array behind the listener. removeListener will remove, at most, one instance of a listener from the listener array. If any single listener has been added multiple times to the listener array for the specified event, then removeListener must be called multiple times to remove each instance. Returns emitter, so calls can be chained.
5	<b>removeAllListeners</b> <i>[event]</i>	Removes all listeners, or those of the specified event. It's not a good idea to remove listeners that were added elsewhere in the code, especially when it's on an emitter that you didn't create <i>e.g. sockets or file streams</i> . Returns emitter, so calls can be chained.
6	<b>setMaxListeners</b> <i>n</i>	By default EventEmitters will print a warning if more than 10 listeners are added for a particular event. This is a useful default which helps finding memory leaks. Obviously not all Emitters should be limited to 10. This function allows that to be increased. Set to zero for unlimited.
7	<b>listeners</b> <i>event</i>	Returns an array of listeners for the specified event.

8

**emitevent,[arg1],[arg2],...]**

Execute each of the listeners in order with the supplied arguments. Returns true if event had listeners, false otherwise.

## Class Methods

Sr. No.	method	Description
1	<b>listenerCount</b> <i>emitter,event</i>	Return the number of listeners for a given event.

## Events

Sr. No.	event name	Parameters	Description
1	<b>newListener</b>	<ul style="list-style-type: none"> <li><b>event</b> - String The event name</li> <li><b>listener</b>- Function The event handler function</li> </ul>	This event is emitted any time a listener is added. When this event is triggered, the listener may not yet have been added to the array of listeners for the event.
2	<b>removeListener</b>	<ul style="list-style-type: none"> <li><b>event</b> - String The event name</li> <li><b>listener</b>- Function The event handler function</li> </ul>	This event is emitted any time someone removes a listener. When this event is triggered, the listener may not yet have been removed from the array of listeners for the event.

## Example

Create a js file named test.js in **C:\>Nodejs\_WorkSpace**.

*File: test.js*

```
var events = require('events');

var eventEmitter = new events.EventEmitter();
```

```

//listener #1
var listner1 = function listner1() {
 console.log('listner1 executed.');
}

//listener #2
var listner2 = function listner2() {
 console.log('listner2 executed.');
}

// bind the connection event with the listner1 function
eventEmitter.addListener('connection', listner1);

// bind the connection event with the listner2 function
eventEmitter.on('connection', listner2);

var eventListeners = require('events').EventEmitter.listenerCount(eventEmitter, 'connection');
console.log(eventListeners + " Listener(s) listening to connection event");

// fire the connection event
eventEmitter.emit('connection');

// remove the binding of listner1 function
eventEmitter.removeListener('connection', listner1);
console.log("Listner1 will not listen now.");

// fire the connection event
eventEmitter.emit('connection');

eventListeners = require('events').EventEmitter.listenerCount(eventEmitter, 'connection');
console.log(eventListeners + " Listener(s) listening to connection event");

console.log("Program Ended.");

```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

Verify the Output. Server has started

```
2 Listener(s) listening to connection event
listner1 executed.
```

```

listner2 executed.
Listner1 will not listen now.
listner2 executed.
1 Listener(s) listening to connection event
Program Ended.

```

## NODE .- BUFFER MODULE

**buffer** module can be used to create Buffer and SlowBuffer classes. Buffer module can be imported using following syntax.

```
var buffer = require("buffer")
```

### Buffer class

Buffer class is a global class and can be accessed in application without importing buffer module. A Buffer is a kind of an array of integers and corresponds to a raw memory allocation outside the V8 heap. A Buffer cannot be resized.

### Methods

Sr. No	method	Parameters	Description
1	<b>new Buffer</b> <i>size</i>	<ul style="list-style-type: none"> <li>• size Number</li> </ul>	Allocates a new buffer of size octets. Note, size must be no more than kMaxLength. Otherwise, a RangeError will be thrown here.
2	<b>new Buffer</b> <i>buffer</i>	<ul style="list-style-type: none"> <li>• buffer Buffer</li> </ul>	Copies the passed buffer data onto a new Buffer instance.
3	<b>new Buffer</b> <i>str[,encoding]</i>	<ul style="list-style-type: none"> <li>• str String - string to encode.</li> <li>• encoding String - encoding to use, Optional.</li> </ul>	Allocates a new buffer containing the given str. encoding defaults to 'utf8'.
4	<b>buf.length</b>	Return: Number	The size of the buffer in bytes. Note that this is not necessarily the size of the contents. length refers to the amount of memory allocated for the buffer object.

			It does not change when the contents of the buffer are changed.
5	<b>buf.writestring[,offset][,length][,encoding]</b>	<ul style="list-style-type: none"> <li>• string String - data to be written to buffer</li> <li>• offset Number, Optional, Default: 0</li> <li>• length Number, Optional, Default: buffer.length - offset</li> <li>• encoding String, Optional, Default: 'utf8'</li> </ul>	Allocates a new buffer containing the given str. encoding defaults to 'utf8'.
6	<b>buf.writeUIntLEvalue,offset,byteLength[,noAssert]</b>	<ul style="list-style-type: none"> <li>• value {Number} Bytes to be written to buffer</li> <li>• offset {Number} <math>0 \leqslant \text{offset} \leqslant \text{buf.length}</math></li> <li>• byteLength {Number} <math>0 &lt; \text{byteLength} \leqslant 6</math></li> <li>• noAssert {Boolean} Default: false</li> <li>• Return: {Number}</li> </ul>	Writes value to the buffer at the specified offset and byteLength. Supports up to 48 bits of accuracy. Set noAssert to true to skip validation of value and offset. Defaults to false.
7	<b>buf.writeUIntBEvalue,offset,byteLength[,noAssert]</b>	<ul style="list-style-type: none"> <li>• value {Number} Bytes to be written to buffer</li> </ul>	Writes value to the buffer at the specified offset and byteLength. Supports up to 48 bits of accuracy. Set noAssert to true

		<ul style="list-style-type: none"> <li>• offset   {Number}   0 &lt;= offset   &lt;= buf.length</li> <li>• byteLength   {Number}   0 &lt;   byteLength   h &lt;= 6</li> <li>• noAssert   {Boolean}   Default:   false</li> <li>• Return:   {Number}</li> </ul>	to skip validation of value and offset. Defaults to false.
8	<b>buf.writeIntLEvalue,offset,byteLength[,noAssert]</b>	<ul style="list-style-type: none"> <li>• value   {Number}   Bytes to be written to buffer</li> <li>• offset   {Number}   0 &lt;= offset   &lt;= buf.length</li> <li>• byteLength   {Number}   0 &lt;   byteLength   h &lt;= 6</li> <li>• noAssert   {Boolean}   Default:   false</li> <li>• Return:   {Number}</li> </ul>	Writes value to the buffer at the specified offset and byteLength. Supports up to 48 bits of accuracy. Set noAssert to true to skip validation of value and offset. Defaults to false.
9	<b>buf.writeIntBEvalue,offset,byteLength[,noAssert]</b>	<ul style="list-style-type: none"> <li>• value   {Number}   Bytes to be written to buffer</li> <li>• offset   {Number}   0 &lt;= offset   &lt;= buf.length</li> <li>• byteLength   {Number}   0 &lt;   byteLength   h &lt;= 6</li> </ul>	Writes value to the buffer at the specified offset and byteLength. Supports up to 48 bits of accuracy. Set noAssert to true to skip validation of value and offset. Defaults to false.

		<ul style="list-style-type: none"> <li>• offset {Number} 0 &lt;= offset &lt;= buf.length</li> <li>• byteLength {Number} 0 &lt; byteLength &lt;= 6</li> <li>• noAssert {Boolean} Default: false</li> <li>• Return: {Number}</li> </ul>	
10	<b>buf.readUIntLEoffset,byteLength[,noAssert]</b>	<ul style="list-style-type: none"> <li>• offset {Number} 0 &lt;= offset &lt;= buf.length</li> <li>• byteLength {Number} 0 &lt; byteLength &lt;= 6</li> <li>• noAssert {Boolean} Default: false</li> <li>• Return: {Number}</li> </ul>	A generalized version of all numeric read methods. Supports up to 48 bits of accuracy. Set noAssert to true to skip validation of offset. This means that offset may be beyond the end of the buffer. Defaults to false.
11	<b>buf.readUIntBEoffset,byteLength[,noAssert]</b>	<ul style="list-style-type: none"> <li>• offset {Number} 0 &lt;= offset &lt;= buf.length</li> <li>• byteLength {Number} 0 &lt; byteLength &lt;= 6</li> <li>• noAssert {Boolean} Default: false</li> <li>• Return: {Number}</li> </ul>	A generalized version of all numeric read methods. Supports up to 48 bits of accuracy. Set noAssert to true to skip validation of offset. This means that offset may be beyond the end of the buffer. Defaults to false.
12	<b>buf.readIntLEoffset,byteLength[,noAssert]</b>	<ul style="list-style-type: none"> <li>• offset {Number} 0 &lt;= offset &lt;= buf.length</li> </ul>	A generalized version of all numeric read methods. Supports up to 48 bits of accuracy. Set noAssert to true to skip validation of offset. This

		<ul style="list-style-type: none"> <li>• byteLength Number 0 &lt; byteLength &lt;= 6</li> <li>• noAssert Boolean Default: false</li> <li>• Return: Number</li> </ul>	means that offset may be beyond the end of the buffer. Defaults to false.
13	<b>buf.readIntBE[offset,byteLength[,noAssert]]</b>	<ul style="list-style-type: none"> <li>• offset Number 0 &lt;= offset &lt;= buf.length</li> <li>• byteLength Number 0 &lt; byteLength &lt;= 6</li> <li>• noAssert Boolean Default: false</li> <li>• Return: Number</li> </ul>	A generalized version of all numeric read methods. Supports up to 48 bits of accuracy. Set noAssert to true to skip validation of offset. This means that offset may be beyond the end of the buffer. Defaults to false.
14	<b>buf.toString[encoding][,start][,end]</b>	<ul style="list-style-type: none"> <li>• encoding String, Optional, Default: 'utf8'</li> <li>• start Number, Optional, Default: 0</li> <li>• end Number, Optional, Default: buffer.length</li> </ul>	Decodes and returns a string from buffer data encoded using the specified character set encoding.
15	<b>buf.toJSON</b>		Returns a JSON-representation of the Buffer instance. JSON.stringify implicitly calls this function when stringifying a Buffer instance.

16	<b>buf[index]</b>		Get and set the octet at index. The values refer to individual bytes, so the legal range is between 0x00 and 0xFF hex or 0 and 255.
17	<b>buf.equals(<i>otherBuffer</i>)</b>	<ul style="list-style-type: none"> <li>• <i>otherBuffer</i> Buffer</li> </ul>	Returns a boolean of whether this and <i>otherBuffer</i> have the same bytes.
18	<b>buf.compare(<i>otherBuffer</i>)</b>	<ul style="list-style-type: none"> <li>• <i>otherBuffer</i> Buffer</li> </ul>	Returns a number indicating whether this comes before or after or is the same as the <i>otherBuffer</i> in sort order.
19	<b>buf.copyWithTargetBuffer[,targetStart][,sourceStart][,sourceEnd]</b>	<ul style="list-style-type: none"> <li>• <i>targetBuffer</i> Buffer object - Buffer to copy into</li> <li>• <i>targetStart</i> Number, Optional, Default: 0</li> <li>• <i>sourceStart</i> Number, Optional, Default: 0</li> <li>• <i>sourceEnd</i> Number, Optional, Default: <i>buffer.length</i></li> </ul>	Copies data from a region of this buffer to a region in the target buffer even if the target memory region overlaps with the source. If undefined the <i>targetStart</i> and <i>sourceStart</i> parameters default to 0 while <i>sourceEnd</i> defaults to <i>buffer.length</i> .
20	<b>buf.slice[<i>start</i>][,<i>end</i>]</b>	<ul style="list-style-type: none"> <li>• <i>start</i> Number, Optional, Default: 0</li> <li>• <i>end</i> Number, Optional, Default: <i>buffer.length</i></li> </ul>	Returns a new buffer which references the same memory as the old, but offset and cropped by the start <i>defaultsto0</i> and end <i>defaultstobuffer.length</i> indexes. Negative indexes start from the end of the buffer.
21	<b>buf.readUInt8(<i>offset</i>)[,noAssert]</b>	<ul style="list-style-type: none"> <li>• <i>offset</i> Number</li> <li>• <i>noAssert</i> Boolean, Optional, Default: false</li> </ul>	Reads an unsigned 8 bit integer from the buffer at the specified offset. Set <i>noAssert</i> to true to skip validation of offset. This means that offset may be beyond the end of the buffer. Defaults to false.

		<ul style="list-style-type: none"> <li>• Return: Number</li> </ul>	
22	<b>buf.readUInt16LEoffset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> <li>• Return: Number</li> </ul>	Reads an unsigned 16 bit integer from the buffer at the specified offset with specified endian format. Set noAssert to true to skip validation of offset. This means that offset may be beyond the end of the buffer. Defaults to false.
23	<b>buf.readUInt16BEoffset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> <li>• Return: Number</li> </ul>	Reads an unsigned 16 bit integer from the buffer at the specified offset with specified endian format. Set noAssert to true to skip validation of offset. This means that offset may be beyond the end of the buffer. Defaults to false.
24	<b>buf.readUInt32LEoffset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> <li>• Return: Number</li> </ul>	Reads an unsigned 32 bit integer from the buffer at the specified offset with specified endian format. Set noAssert to true to skip validation of offset. This means that offset may be beyond the end of the buffer. Defaults to false.
25	<b>buf.readUInt32BEoffset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> <li>• Return: Number</li> </ul>	Reads an unsigned 32 bit integer from the buffer at the specified offset with specified endian format. Set noAssert to true to skip validation of offset. This means that offset may be beyond the end of the buffer. Defaults to false.
26	<b>buf.readInt8offset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> </ul>	Reads a signed 8 bit integer from the buffer at the specified offset. Set noAssert to true to skip validation of offset. This means that offset may be beyond the end of the buffer. Defaults to false.

		<ul style="list-style-type: none"> <li>• Return: Number</li> </ul>	
27	<b>buf.readInt16LEoffset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> <li>• Return: Number</li> </ul>	Reads a signed 16 bit integer from the buffer at the specified offset with specified endian format. Set noAssert to true to skip validation of offset. This means that offset may be beyond the end of the buffer. Defaults to false.
28	<b>buf.readInt16BEoffset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> <li>• Return: Number</li> </ul>	Reads a signed 16 bit integer from the buffer at the specified offset with specified endian format. Set noAssert to true to skip validation of offset. This means that offset may be beyond the end of the buffer. Defaults to false.
29	<b>buf.readInt32LEoffset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> <li>• Return: Number</li> </ul>	Reads a signed 32 bit integer from the buffer at the specified offset with specified endian format. Set noAssert to true to skip validation of offset. This means that offset may be beyond the end of the buffer. Defaults to false.
30	<b>buf.readInt32BEoffset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> <li>• Return: Number</li> </ul>	Reads a signed 32 bit integer from the buffer at the specified offset with specified endian format. Set noAssert to true to skip validation of offset. This means that offset may be beyond the end of the buffer. Defaults to false.
31	<b>buf.readFloatLEoffset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> </ul>	Reads a 32 bit float from the buffer at the specified offset with specified endian format. Set noAssert to true to skip validation of offset. This means that offset may be beyond the

		<ul style="list-style-type: none"> <li>• Return: Number</li> </ul>	end of the buffer. Defaults to false.
32	<b>buf.readFloatBEoffset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> <li>• Return: Number</li> </ul>	Reads a 32 bit float from the buffer at the specified offset with specified endian format. Set noAssert to true to skip validation of offset. This means that offset may be beyond the end of the buffer. Defaults to false.
33	<b>buf.readDoubleLEoffset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> <li>• Return: Number</li> </ul>	Reads a 64 bit double from the buffer at the specified offset with specified endian format. Set noAssert to true to skip validation of offset. This means that offset may be beyond the end of the buffer. Defaults to false.
34	<b>buf.readDoubleBEoffset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> <li>• Return: Number</li> </ul>	Reads a 64 bit double from the buffer at the specified offset with specified endian format. Set noAssert to true to skip validation of offset. This means that offset may be beyond the end of the buffer. Defaults to false.
35	<b>buf.writeUInt8value,offset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• value Number</li> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> </ul>	Writes value to the buffer at the specified offset. Note, value must be a valid unsigned 8 bit integer. Set noAssert to true to skip validation of value and offset. This means that value may be too large for the specific function and offset may be beyond the end of the buffer leading to the values being silently dropped. This should not be used unless you are certain of correctness. Defaults to false.
36	<b>buf.writeUInt16LEvalue,offset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• value Number</li> </ul>	Writes value to the buffer at the specified offset with specified endian format. Note, value must

		<ul style="list-style-type: none"> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> </ul>	be a valid unsigned 16 bit integer. Set noAssert to true to skip validation of value and offset. This means that value may be too large for the specific function and offset may be beyond the end of the buffer leading to the values being silently dropped. This should not be used unless you are certain of correctness. Defaults to false.
37	<b>buf.writeUInt16BEvalue,offset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• value Number</li> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> </ul>	Writes value to the buffer at the specified offset with specified endian format. Note, value must be a valid unsigned 16 bit integer. Set noAssert to true to skip validation of value and offset. This means that value may be too large for the specific function and offset may be beyond the end of the buffer leading to the values being silently dropped. This should not be used unless you are certain of correctness. Defaults to false.
38	<b>buf.writeUInt32LEvalue,offset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• value Number</li> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> </ul>	Writes value to the buffer at the specified offset with specified endian format. Note, value must be a valid unsigned 32 bit integer. Set noAssert to true to skip validation of value and offset. This means that value may be too large for the specific function and offset may be beyond the end of the buffer leading to the values being silently dropped. This should not be used unless you are certain of correctness. Defaults to false.
39	<b>buf.writeUInt32BEvalue,offset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• value Number</li> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> </ul>	Writes value to the buffer at the specified offset with specified endian format. Note, value must be a valid unsigned 32 bit integer. Set noAssert to true to skip validation of value and offset. This means that value may be too large for the specific function and offset may be beyond the end of the buffer

			leading to the values being silently dropped. This should not be used unless you are certain of correctness. Defaults to false.
40	<b>buf.writeInt8</b> <i>value,offset[,noAssert]</i>	<ul style="list-style-type: none"> <li>• value Number</li> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> </ul>	Writes value to the buffer at the specified offset with specified endian format. Note, value must be a valid signed 8 bit integer. Set noAssert to true to skip validation of value and offset. This means that value may be too large for the specific function and offset may be beyond the end of the buffer leading to the values being silently dropped. This should not be used unless you are certain of correctness. Defaults to false.
41	<b>buf.writeInt16LE</b> <i>value,offset[,noAssert]</i>	<ul style="list-style-type: none"> <li>• value Number</li> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> </ul>	Writes value to the buffer at the specified offset with specified endian format. Note, value must be a valid signed 16 bit integer. Set noAssert to true to skip validation of value and offset. This means that value may be too large for the specific function and offset may be beyond the end of the buffer leading to the values being silently dropped. This should not be used unless you are certain of correctness. Defaults to false.
42	<b>buf.writeInt16BE</b> <i>value,offset[,noAssert]</i>	<ul style="list-style-type: none"> <li>• value Number</li> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> </ul>	Writes value to the buffer at the specified offset with specified endian format. Note, value must be a valid signed 16 bit integer. Set noAssert to true to skip validation of value and offset. This means that value may be too large for the specific function and offset may be beyond the end of the buffer leading to the values being silently dropped. This should not be used unless you are certain of correctness. Defaults to false.

43	<b>buf.writeInt32LEvalue,offset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• value Number</li> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> </ul>	Writes value to the buffer at the specified offset with specified endian format. Note, value must be a valid signed 32 bit integer. Set noAssert to true to skip validation of value and offset. This means that value may be too large for the specific function and offset may be beyond the end of the buffer leading to the values being silently dropped. This should not be used unless you are certain of correctness. Defaults to false.
44	<b>buf.writeInt32BEvalue,offset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• value Number</li> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> </ul>	Writes value to the buffer at the specified offset with specified endian format. Note, value must be a valid signed 32 bit integer. Set noAssert to true to skip validation of value and offset. This means that value may be too large for the specific function and offset may be beyond the end of the buffer leading to the values being silently dropped. This should not be used unless you are certain of correctness. Defaults to false.
45	<b>buf.writeFloatLEvalue,offset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• value Number</li> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> </ul>	Writes value to the buffer at the specified offset with specified endian format. Note, value must be a valid 32 bit float. Set noAssert to true to skip validation of value and offset. This means that value may be too large for the specific function and offset may be beyond the end of the buffer leading to the values being silently dropped. This should not be used unless you are certain of correctness. Defaults to false.
46	<b>buf.writeFloatBEvalue,offset[,noAssert]</b>	<ul style="list-style-type: none"> <li>• value Number</li> <li>• offset Number</li> <li>• noAssert Boolean, Optional,</li> </ul>	Writes value to the buffer at the specified offset with specified endian format. Note, value must be a valid 32 bit float. Set noAssert to true to skip validation of value and offset. This means that value may be

		Default: false	too large for the specific function and offset may be beyond the end of the buffer leading to the values being silently dropped. This should not be used unless you are certain of correctness. Defaults to false.
47	<b>buf.writeDoubleLE</b> <i>value,offset[,noAssert]</i>	<ul style="list-style-type: none"> <li>• value Number</li> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> </ul>	Writes value to the buffer at the specified offset with specified endian format. Note, value must be a valid 64 bit double. Set noAssert to true to skip validation of value and offset. This means that value may be too large for the specific function and offset may be beyond the end of the buffer leading to the values being silently dropped. This should not be used unless you are certain of correctness. Defaults to false.
48	<b>buf.writeDoubleBE</b> <i>value,offset[,noAssert]</i>	<ul style="list-style-type: none"> <li>• value Number</li> <li>• offset Number</li> <li>• noAssert Boolean, Optional, Default: false</li> </ul>	Writes value to the buffer at the specified offset with specified endian format. Note, value must be a valid 64 bit double. Set noAssert to true to skip validation of value and offset. This means that value may be too large for the specific function and offset may be beyond the end of the buffer leading to the values being silently dropped. This should not be used unless you are certain of correctness. Defaults to false.
49	<b>buf.fill</b> <i>value[,offset][,end]</i>	<ul style="list-style-type: none"> <li>• value Number</li> <li>• offset Number, Optional</li> <li>• end Number, Optional</li> </ul>	Fills the buffer with the specified value. If the offset <i>defaultsto 0</i> and end <i>defaultsto buffer.length</i> are not given it will fill the entire buffer.

## Class Methods

Sr. No.	method	Parameters	Description
1	<b>Buffer.isEncodingencoding</b>	<ul style="list-style-type: none"> <li>• encoding String The encoding string to test</li> </ul>	Returns true if the encoding is a valid encoding argument, or false otherwise.
2	<b>Buffer.isBufferobj</b>	<ul style="list-style-type: none"> <li>• obj Object</li> <li>• Return: Boolean</li> </ul>	Tests if obj is a Buffer.
3	<b>Buffer.byteLengthstring[,encoding]</b>	<ul style="list-style-type: none"> <li>• string String</li> <li>• encoding String, Optional, Default: 'utf8'</li> <li>• Return: Number</li> </ul>	Gives the actual byte length of a string. encoding defaults to 'utf8'. This is not the same as String.prototype.length since that returns the number of characters in a string.
4	<b>Buffer.concatlist[,totalLength]</b>	<ul style="list-style-type: none"> <li>• list Array List of Buffer objects to concat</li> <li>• totalLength Number Total length of the buffers when concatenated</li> </ul>	Returns a buffer which is the result of concatenating all the buffers in the list together.
5	<b>Buffer.comparebuf1,buf2</b>	<ul style="list-style-type: none"> <li>• buf1 Buffer</li> <li>• buf2 Buffer</li> </ul>	The same as buf1.comparebuf2. Useful for sorting an Array of Buffers.

## PROTOTYPE

```

function Gamer(){
 this.name = "",
 this.life= 100,
 this.gavelife= function give(opponent)
 {
 opponent.life += 1;
 console.log(this.name+" gave 1 life to "+opponent.name);
 }
}
var Y = new Gamer();
var I = new Gamer();
Y.name = "Yunus";
I.name = "Irshad";
Y.gavelife(I);
console.log("Yunus: "+Y.life);
console.log("Irshad: "+I.life);

// You can add functions to all objects....
Gamer.prototype.minus = function minus(opponent)
{
 opponent.life -= 3;
 console.log(this.name+" minus 3 life from "+opponent.name);
}
I_MINUS(Y);
console.log("Yunus: "+Y.life);
console.log("Irshad: "+I.life);

```

## Example

Create a js file named test.js in C:\>Nodejs\_WorkSpace.

File: test.js

```
//create a buffer
var buffer = new Buffer(26);
```

```

// You can add properties to all objects.....
Gamer.prototype.propertyName=120;
console.log(Y.propertyName);
console.log(I.propertyName);

```

```
console.log("buffer length: " + buffer.length);

//write to buffer
var data = "TutorialsPoint.com";
buffer.write(data);

console.log(data + ": " + data.length + " characters, " + Buffer.byteLength(data, 'utf8') + " bytes");

//slicing a buffer
var buffer1 = buffer.slice(0,14);
console.log("buffer1 length: " + buffer1.length);
console.log("buffer1 content: " + buffer1.toString());

//modify buffer by indexes
for (var i = 0 ; i < 26 ; i++) {
 buffer[i] = i + 97; // 97 is ASCII a
}
console.log("buffer content: " + buffer.toString('ascii'));

var buffer2 = new Buffer(4);

buffer2[0] = 0x3;
buffer2[1] = 0x4;
buffer2[2] = 0x23;
buffer2[3] = 0x42;

//reading from buffer
console.log(buffer2.readUInt16BE(0));
console.log(buffer2.readUInt16LE(0));
console.log(buffer2.readUInt16BE(1));
console.log(buffer2.readUInt16LE(1));
console.log(buffer2.readUInt16BE(2));
console.log(buffer2.readUInt16LE(2));

var buffer3 = new Buffer(4);
buffer3.writeUInt16BE(0xdead, 0);
buffer3.writeUInt16BE(0xbeef, 2);

console.log(buffer3);
```

```

buffer3.writeUInt16LE(0xdead, 0);

buffer3.writeUInt16LE(0xbeef, 2);

console.log(buffer3);
//convert to a JSON Object
var json = buffer3.toJSON();
console.log("JSON Representation : ");
console.log(json);

//Get a buffer from JSON Object
var buffer6 = new Buffer(json);
console.log(buffer6);

//copy a buffer
var buffer4 = new Buffer(26);
buffer.copy(buffer4);
console.log("buffer4 content: " + buffer4.toString());

//concatenate a buffer
var buffer5 = Buffer.concat([buffer, buffer4]);
console.log("buffer5 content: " + buffer5.toString());

```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

Verify the Output.

```

buffer length: 26
TutorialsPoint.com: 18 characters, 18 bytes
buffer1 length: 14
buffer1 content: TutorialsPoint
buffer content: abcdefghijklmnopqrstuvwxyz
772
1027
1059
8964
9026
16931
<Buffer de ad be ef>
<Buffer ad de ef be>
buffer4 content: abcdefghijklmnopqrstuvwxyz
buffer5 content: abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz

```

## NODE .- STREAMS

---

**What are Streams?**

Streams are objects that let you read data from a source or write data to destination in continuous fashion. In Node, there are four types of streams.

- **Readable** - Stream which is used for read operation.
- **Writable** - Stream which is used for write operation.
- **Duplex** - Stream which can be used for both read and write operation.
- **Transform** - A type of duplex stream where the output is computed based on input.

Each type of Stream is an EventEmitter and throws several events at times. For example, some of the commonly used events are:

- **data** - This event is fired when there is data is available to read.
- **end** - This event is fired when there is no more data to read.
- **error** - This event is fired when there is any error receiving or writing data.
- **finish** - This event is fired when all data has been flushed to underlying system

## Reading from stream

Create a txt file named test.txt in **C:\>Nodejs\_WorkSpace**

TutorialsPoint.Com

Create test.js in **C:\>Nodejs\_WorkSpace**

```
var fs = require("fs");
var data = '';
//create a readable stream
var readerStream = fs.createReadStream('test.txt');

//set the encoding to be utf8.
readerStream.setEncoding('UTF8');

//handle stream events
readerStream.on('data', function(chunk) {
 data += chunk;
});

readerStream.on('end', function() {
 console.log(data);
});

readerStream.on('error', function(err) {
 console.log(err.stack);
});
console.log("Program Ended");
```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

## Verify the Output

```
Program Ended
TutorialsPoint.Com
```

## Writing to stream

Update test.js in C:\>Nodejs\_WorkSpace

```
var fs = require("fs");
var data = 'TutorialsPoint.Com';
//create a writable stream
var writerStream = fs.createWriteStream('test1.txt');

//write the data to stream
//set the encoding to be utf8.
writerStream.write(data, 'UTF8');

//mark the end of file
writerStream.end();

//handle stream events
writerStream.on('finish', function() {
 console.log("Write completed.");
});

writerStream.on('error', function(err){
 console.log(err.stack);
});
console.log("Program Ended");
```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

## Verify the Output

```
Program Ended
Write completed.
```

Open test1.txt in C:\>Nodejs\_WorkSpace.Verify the result.

## Piping streams

Piping is a mechanism to connect output of one stream to another stream. It is normally used to get data from one stream and to pass output of that stream to another stream. There is no limit on piping operations. Consider the above example, where we've read test.txt using readerStream and write test1.txt using writerStream. Now we'll use the piping to simplify our operation or reading from one file and writing to another file.

Update test.js in **C:\>Nodejs\_WorkSpace**

```
var fs = require("fs");

//create a readable stream
var readerStream = fs.createReadStream('test.txt');

//create a writable stream
var writerStream = fs.createWriteStream('test2.txt');

//pipe the read and write operations
//read test.txt and write data to test2.txt
readerStream.pipe(writerStream);

console.log("Program Ended");
```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

Verify the Output

```
Program Ended
```

Open test2.txt in **C:\>Nodejs\_WorkSpace**. Verify the result.

---

N O D E . - H I S L E     S Y S T E M

**fs** module is used for File I/O. **fs** module can be imported using following syntax.

```
var fs = require("fs")
```

**Synchronous vs Asynchronous**

Every method in fs module have synchronous as well as asynchronous form. Asynchronous methods takes a last parameter as completion function callback and first parameter of the callback function is error. It is preferred to use asynchronous method instead of synchronous method as former never block the program execution where the latter one does.

## Example

Create a js file named test.js in **C:\>Nodejs\_WorkSpace**.

*File: test.js*

```
var fs = require("fs");

//Asynchronous read
fs.readFile('test.txt', function (err, data) {
 if (err) return console.error(err);
 console.log("Asynchronous read: " + data.toString());
});

//Synchronous read
var data = fs.readFileSync('test.txt');
console.log("Synchronous read: " + data.toString());

console.log("Program Ended");
```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

Verify the Output.

```
Synchronous read: TutorialsPoint.Com
Program Ended
Asynchronous read: TutorialsPoint.Com
```

## Methods

Sr. No.	method	Description
1	<b>fs.renameoldPath,newPath,callback</b>	Asynchronous rename. No arguments other than a possible exception are given to the completion callback.

2	<b>fs.ftruncatefd,len,callback</b>	Asynchronous ftruncate. No arguments other than a possible exception are given to the completion callback.
3	<b>fs.ftruncateSyncfd,len</b>	Synchronous ftruncate
4	<b>fs.truncatepath,len,callback</b>	Asynchronous truncate. No arguments other than a possible exception are given to the completion callback.
5	<b>fs.truncateSyncpath,len</b>	Synchronous truncate
6	<b>fs.chownpath,uid,gid,callback</b>	Asynchronous chown. No arguments other than a possible exception are given to the completion callback.
7	<b>fs.chownSyncpath,uid,gid</b>	Synchronous chown
8	<b>fs.fchownfd,uid,gid,callback</b>	Asynchronous fchown. No arguments other than a possible exception are given to the completion callback.
9	<b>fs.fchownSyncfd,uid,gid</b>	Synchronous fchown
10	<b>fs.lchownpath,uid,gid,callback</b>	Asynchronous lchown. No arguments other than a possible exception are given to the completion callback.
11	<b>fs.lchownSyncpath,uid,gid</b>	Synchronous lchown
12	<b>fs.chmodpath,mode,callback</b>	Asynchronous chmod. No arguments other than a possible exception are given to the completion callback.
13	<b>fs.chmodSyncpath,mode</b>	Synchronous chmod.
14	<b>fs.fchmodfd,mode,callback</b>	Asynchronous fchmod. No arguments other than a possible exception are given to the completion callback.
15	<b>fs.fchmodSyncfd,mode</b>	Synchronous fchmod.
16	<b>fs.lchmodpath,mode,callback</b>	Asynchronous lchmod. No arguments other than a possible exception are given to the completion callback. Only available on Mac OS X.
17	<b>fs.lchmodSyncpath,mode</b>	Synchronous lchmod.

18	<b>fs.statpath,callback</b>	Asynchronous stat. The callback gets two arguments <i>err,stats</i> where stats is a fs.Stats object.
19	<b>fs.lstatpath,callback</b>	Asynchronous lstat. The callback gets two arguments <i>err,stats</i> where stats is a fs.Stats object. lstat is identical to stat, except that if path is a symbolic link, then the link itself is stat-ed, not the file that it refers to.
20	<b>fs.fstatfd,callback</b>	Asynchronous fstat. The callback gets two arguments <i>err,stats</i> where stats is a fs.Stats object. fstat is identical to stat, except that the file to be stat-ed is specified by the file descriptor fd.
21	<b>fs.statSyncpath</b>	Synchronous stat. Returns an instance of fs.Stats.
22	<b>fs.lstatSyncpath</b>	Synchronous lstat. Returns an instance of fs.Stats.
23	<b>fs.fstatSyncfd</b>	Synchronous fstat. Returns an instance of fs.Stats.
24	<b>fs.linksrcpath,dstpath,callback</b>	Asynchronous link. No arguments other than a possible exception are given to the completion callback.
25	<b>fs.linkSyncsrcpath,dstpath</b>	Synchronous link.
26	<b>fs.symlinksrcpath,dstpath[,type],callback</b>	Asynchronous symlink. No arguments other than a possible exception are given to the completion callback. The type argument can be set to 'dir', 'file', or 'junction' <i>default is 'file'</i> and is only available on Windows <i>ignored on other platforms</i> . Note that Windows junction points require the destination path to be absolute. When using 'junction', the destination argument will automatically be normalized to absolute path.
27	<b>fs.symlinkSyncsrcpath,dstpath[,type]</b>	Synchronous symlink.
28	<b>fs.readlinkpath,callback</b>	Asynchronous readlink. The callback gets two arguments <i>err,linkString</i> .

29	<b>fs.realpath</b> <i>path[,cache],callback</i>	Asynchronous realpath. The callback gets two arguments <i>err,resolvedPath</i> . May use process.cwd to resolve relative paths. cache is an object literal of mapped paths that can be used to force a specific path resolution or avoid additional fs.stat calls for known real paths.
30	<b>fs.realpathSync</b> <i>path[,cache]</i>	Synchronous realpath. Returns the resolved path.
31	<b>fs.unlink</b> <i>path,callback</i>	Asynchronous unlink. No arguments other than a possible exception are given to the completion callback.
32	<b>fs.unlinkSync</b> <i>path</i>	Synchronous unlink.
33	<b>fs.rmdir</b> <i>path,callback</i>	Asynchronous rmdir. No arguments other than a possible exception are given to the completion callback.
34	<b>fs.rmdirSync</b> <i>path</i>	Synchronous rmdir.
35	<b>fs.mkdir</b> <i>path[,mode],callback</i>	Asynchronous mkdir2. No arguments other than a possible exception are given to the completion callback. mode defaults to 0777.
36	<b>fs.mkdirSync</b> <i>path[,mode]</i>	Synchronous mkdir.
37	<b>fs.readdir</b> <i>path,callback</i>	Asynchronous readdir3. Reads the contents of a directory. The callback gets two arguments <i>err,files</i> where files is an array of the names of the files in the directory excluding '.' and '..'.
38	<b>fs.readdirSync</b> <i>path</i>	Synchronous readdir. Returns an array of filenames excluding '.' and '..'.
39	<b>fs.close</b> <i>fd,callback</i>	Asynchronous close. No arguments other than a possible exception are given to the completion callback.
40	<b>fs.closeSync</b> <i>fd</i>	Synchronous close.
41	<b>fs.open</b> <i>path,flags[,mode],callback</i>	Asynchronous file open.
42	<b>fs.openSync</b> <i>path,flags[,mode]</i>	Synchronous version of fs.open.
43	<b>fs.utimes</b> <i>path,atime,mtime,callback</i>	

44	<b>fs.utimesSyncpath,atime,mtime</b>	Change file timestamps of the file referenced by the supplied path.
45	<b>fs.futimesfd,atime,mtime,callback</b>	
46	<b>fs.futimesSyncfd,atime,mtime</b>	Change the file timestamps of a file referenced by the supplied file descriptor.
47	<b>fs.fsyncfd,callback</b>	Asynchronous fsync2. No arguments other than a possible exception are given to the completion callback.
48	<b>fs.fsyncSyncfd</b>	Synchronous fsync2.
49	<b>fs.writefd,buffer,offset,length[,position],callback</b>	Write buffer to the file specified by fd.
50	<b>fs.writefd,data[,position[,encoding]],callback</b>	Write data to the file specified by fd. If data is not a Buffer instance then the value will be coerced to a string.
51	<b>fs.writeSyncfd,buffer,offset,length[,position]</b>	Synchronous versions of fs.write. Returns the number of bytes written.
52	<b>fs.writeSyncfd,data[,position[,encoding]]</b>	Synchronous versions of fs.write. Returns the number of bytes written.
53	<b>fs.readfd,buffer,offset,length,position,callback</b>	Read data from the file specified by fd.
54	<b>fs.readSyncfd,buffer,offset,length,position</b>	Synchronous version of fs.read. Returns the number of bytesRead.
55	<b>fs.readFilefilename[,options],callback</b>	Asynchronously reads the entire contents of a file.
56	<b>fs.readFileSyncfilename[,options]</b>	Synchronous version of fs.readFile. Returns the contents of the filename.
57	<b>fs.writeFilefilename,data[,options],callback</b>	Asynchronously writes data to a file, replacing the file if it already exists. data can be a string or a buffer.
58	<b>fs.writeFileSyncfilename,data[,options]</b>	The synchronous version of fs.writeFile.  <code>var fs = require('fs'); // single quotations and variable name = core module name</code>
59	<b>fs.appendFilefilename,data[,options],callback</b>	Asynchronously append data to a file, creating the file if it not yet  <code>fs.writeFileSync("Yunus.txt", " Yunus is a very good boy");  console.log(fs.readFileSync("Yunus.txt").toString()); // display in terminal</code>

		exists. data can be a string or a buffer.
60	<b>fs.appendFileSync</b> <i>filename,data[,options]</i>	The synchronous version of fs.appendFile.
61	<b>fs.watchFile</b> <i>filename[,options],listener</i>	Watch for changes on filename. The callback listener will be called each time the file is accessed.
62	<b>fs.unwatchFile</b> <i>filename[,listener]</i>	Stop watching for changes on filename. If listener is specified, only that particular listener is removed. Otherwise, all listeners are removed and you have effectively stopped watching filename.
63	<b>fs.watch</b> <i>filename[,options][,listener]</i>	Watch for changes on filename, where filename is either a file or a directory. The returned object is a fs.FSWatcher.
64	<b>fs.exists</b> <i>path,callback</i>	Test whether or not the given path exists by checking with the file system. Then call the callback argument with either true or false.
65	<b>fs.existsSync</b> <i>path</i>	Synchronous version of fs.exists.
66	<b>fs.access</b> <i>path[,mode],callback</i>	Tests a user's permissions for the file specified by path. mode is an optional integer that specifies the accessibility checks to be performed.
67	<b>fs.accessSync</b> <i>path[,mode]</i>	Synchronous version of fs.access. This throws if any accessibility checks fail, and does nothing otherwise.
68	<b>fs.createReadStream</b> <i>path[,options]</i>	Returns a new ReadStream object.
69	<b>fs.createWriteStream</b> <i>path[,options]</i>	Returns a new WriteStream object.
70	<b>fs.symlink</b> <i>srcpath,dstpath[,type],callback</i>	Asynchronous symlink. No arguments other than a possible exception are given to the completion callback. The type argument can be set to 'dir', 'file', or 'junction' <i>default is 'file'</i> and is only available on Windows <i>ignored on other platforms</i> . Note that Windows junction points require the destination path to be absolute. When using 'junction', the

destination argument will automatically be normalized to absolute path.

## Flags

flags for read/write operations are:

- **r** - Open file for reading. An exception occurs if the file does not exist.
- **r+** - Open file for reading and writing. An exception occurs if the file does not exist.
- **rs** - Open file for reading in synchronous mode. Instructs the operating system to bypass the local file system cache. This is primarily useful for opening files on NFS mounts as it allows you to skip the potentially stale local cache. It has a very real impact on I/O performance so don't use this flag unless you need it. Note that this doesn't turn `fs.open` into a synchronous blocking call. If that's what you want then you should be using `fs.openSync`
- **rs+** - Open file for reading and writing, telling the OS to open it synchronously. See notes for 'rs' about using this with caution.
- **w** - Open file for writing. The file is created *if it does not exist* or truncated *if it exists*.
- **wx** - Like 'w' but fails if path exists.
- **w+** - Open file for reading and writing. The file is created *if it does not exist* or truncated *if it exists*.
- **wx+** - Like 'w+' but fails if path exists.
- **a** - Open file for appending. The file is created if it does not exist.
- **ax** - Like 'a' but fails if path exists.
- **a+** - Open file for reading and appending. The file is created if it does not exist.
- **ax'** - Like 'a+' but fails if path exists.

## Example

Create a txt file named test.txt in **C:\>Nodejs\_WorkSpace**

TutorialsPoint.Com

Create a js file named test.js in **C:\>Nodejs\_WorkSpace**.

*File: test.js*

```
var fs = require("fs");
var buffer = new Buffer(1024);

//Example: Opening File
function openFile() {
 console.log("\nOpen file");
 fs.open('test.txt', 'r+', function(err, fd) {
 if (err) console.log(err.stack);
 console.log("File opened");
 });
}
```

```
//Example: Getting File Info

function getStats() {
 console.log("\nGetting File Info");
 fs.stat('test.txt', function (err, stats) {
 if (err) console.log(err.stack);
 console.log(stats);
 console.log("isFile ? "+stats.isFile());
 console.log("isDirectory ? "+stats.isDirectory());
 });
}

//Example: Writing File

function writeFile() {
 console.log("\nWrite file");
 fs.open('test1.txt', 'w+', function(err,fd) {
 var data = "TutorialsPoint.com - Simply Easy Learning!";
 buffer.write(data);

 fs.write(fd, buffer,0,data.length,0,function(err, bytes) {
 if (err) console.log(err.stack);
 console.log(bytes + " written!");
 });
 });
}

//Example: Read File

function readFile() {
 console.log("\nRead file");
 fs.open('test1.txt', 'r+', function(err,fd) {
 if (err) console.log(err.stack);
 fs.read(fd, buffer,0,buffer.length,0,function(err, bytes) {
 if (err) console.log(err.stack);
 console.log(bytes + " read!");
 if(bytes > 0){
 console.log(buffer.slice(0,bytes).toString());
 }
 });
 });
}
```

```

function closeFile() {
 console.log("\nClose file");
 fs.open('test.txt', 'r+', function(err,fd) {
 if (err) console.log(err.stack);
 fs.close(fd, function() {
 if (err) console.log(err.stack);
 console.log("File closed!");
 });
 });
}

function deleteFile() {
 console.log("\nDelete file");
 fs.open('test1.txt', 'r+', function(err,fd) {
 fs.unlink('test1.txt', function(err) {
 if (err) console.log(err.stack);
 console.log("File deleted!");
 });
 });
}

function truncateFile() {
 console.log("\nTruncate file");
 fs.open('test.txt', 'r+', function(err,fd) {
 fs.ftruncate(fd, function(err) {
 if (err) console.log(err.stack);
 console.log("File truncated!");
 });
 });
}

function createDirectory() {
 console.log("\nCreate Directory");
 fs.mkdir('test', function(err) {
 if(!err) {
 console.log("Directory created!");
 }
 == compares only the values.....
 if(err && err.code === 'EXIST') {
 console.log("Directory exists!");
 } else if (err) {
 console.log(err.stack);
 }
 === compares values and data types also
 });
}

```

```
}

}) ;

}

function removeDirectory() {
 console.log("\nRemove Directory");
 fs.rmdir('test', function(err) {
 if(!err) {
 console.log("Directory removed!");
 }
 if (err) {
 console.log("Directory do not exist!");
 }
 });
}

function watchFile() {
 fs.watch('test.txt', function (event, filename) {
 console.log('event is: ' + event);
 });
}

//Opening file
openFile();

//Writing File
writeFile();

//Reading File
readFile();

//Closing Files
closeFile();

//Getting file information
getStats();

//Deleting Files
deleteFile();

//Truncating Files
```

```
truncateFile();

//Creating Directories
createDirectory();

//Removing Directories
removeDirectory();

//Watching File Changes
watchFile();
```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

Verify the Output.

```
Open file
Write file
Read file
Close file
Getting File Info
Delete file
Truncate file
Create Directory
Remove Directory
File opened
{ dev: 0,
 mode: 33206,
 nlink: 1,
 uid: 0,
 gid: 0,
 rdev: 0,
 ino: 0,
 size: 0,
 atime: Fri Jan 01 2010 00:02:15 GMT+0530 (India Standard Time),
 mtime: Sun Feb 15 2015 13:33:09 GMT+0530 (India Standard Time),
 ctime: Fri Jan 01 2010 00:02:15 GMT+0530 (India Standard Time) }
.isFile ? true
.isDirectory ? false
Directory created!
Directory removed!
event is: rename
event is: rename
42 written!
42 read!
TutorialsPoint.com - Simply Easy Learning!
File closed!
File deleted!
```

## NODE .-UTILITY MODULES

In this article, we'll discuss some of the utility modules provided by Node.js library which are very common and are frequently used across the applications.

Sr.No.	Module Name & Description
1	<b>Console</b> Used to print information on stdout and stderr.
2	<b>Process</b> Used to get information on current process. Provides multiple events related to process activities.
3	<b>OS Module</b> Provides basic operating-system related utility functions.
4	<b>Path Module</b> Provides utilities for handling and transforming file paths.
5	<b>Net Module</b> Provides both servers and clients as streams. Acts as a network wrapper.
6	<b>DNS Module</b> Provides functions to do actual DNS lookup as well as to use underlying operating system name resolution functionalities.

**Domain Module**

Provides way to handle multiple different I/O operations as a single group.

## N O D E . - C O N S O L E

---

**console** is a global object and is used to print to stdout and stderr. It is used in synchronous way when destination is file or a terminal and asynchronous way when destination is a pipe.

### Methods

Sr. No.	method	Description
1	<b>console.log</b> [ <i>data</i> ][,...]	Prints to stdout with newline. This function can take multiple arguments in a printf-like way.
2	<b>console.info</b> [ <i>data</i> ][,...]	Prints to stdout with newline. This function can take multiple arguments in a printf-like way.
3	<b>console.error</b> [ <i>data</i> ][,...]	Prints to stderr with newline. This function can take multiple arguments in a printf-like way.
4	<b>console.warn</b> [ <i>data</i> ][,...]	Prints to stderr with newline. This function can take multiple arguments in a printf-like way
5	<b>console.dir</b> <i>obj</i> [, <i>options</i> ]	Uses util.inspect on <i>obj</i> and prints resulting string to stdout.
6	<b>console.time</b> <i>label</i>	Mark a time.
7	<b>console.timeEnd</b> <i>label</i>	Finish timer, record output.
8	<b>console.trace</b> <i>message</i> [,...]	Print to stderr 'Trace :', followed by the formatted message and stack trace to the current position.
9	<b>console.assert</b> <i>value</i> [, <i>message</i> ][,...]	Similar to assert.ok, but the error message is formatted as util.formatmessage....

### Example

Create a js file named test.js in **C:\>Nodejs\_WorkSpace**.

*File: test.js*

```

var counter = 10;

console.log("Counter: %d", counter);

console.time("Getting data");
//make a database call to retrive the data
//getDataFromDataBase();
console.timeEnd('Getting data');

console.info("Program Ended!")

```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

Verify the Output.

```

Counter: 10
Getting data: 0ms
Program Ended!

```

## N O D E . - P R O C E S S

---

**process** is a global object and is used to represent Node process.

### Exit Codes

Node normally exit with a 0 status code when no more async operations are pending. There are other exit codes which are described below:

Code	Name	Description
1	<b>Uncaught Fatal Exception</b>	There was an uncaught exception, and it was not handled by a domain or an uncaughtException event handler.
2	<b>Unused</b>	reserved by Bash for builtin misuse

3	<b>Internal JavaScript Parse Error</b>	The JavaScript source code internal in Node's bootstrapping process caused a parse error. This is extremely rare, and generally can only happen during development of Node itself.
4	<b>Internal JavaScript Evaluation Failure</b>	The JavaScript source code internal in Node's bootstrapping process failed to return a function value when evaluated. This is extremely rare, and generally can only happen during development of Node itself.
5	<b>Fatal Error</b>	There was a fatal unrecoverable error in V8. Typically a message will be printed to stderr with the prefix FATAL ERROR.
6	<b>Non-function Internal Exception Handler</b>	There was an uncaught exception, but the internal fatal exception handler function was somehow set to a non-function, and could not be called.
7	<b>Internal Exception Handler Run-Time Failure</b>	There was an uncaught exception, and the internal fatal exception handler function itself threw an error while attempting to handle it.
8	<b>Unused</b>	
9	<b>Invalid Argument</b>	Either an unknown option was specified, or an option requiring a value was provided without a value.

10	<b>Internal JavaScript Run-Time Failure</b>	The JavaScript source code internal in Node's bootstrapping process threw an error when the bootstrapping function was called. This is extremely rare, and generally can only happen during development of Node itself.
12	<b>Invalid Debug Argument</b>	The --debug and/or --debug-brk options were set, but an invalid port number was chosen.
>128	<b>Signal Exits</b>	If Node receives a fatal signal such as SIGKILL or SIGHUP, then its exit code will be 128 plus the value of the signal code. This is a standard Unix practice, since exit codes are defined to be 7-bit integers, and signal exits set the high-order bit, and then contain the value of the signal code.

## Events

Process is an eventEmitter and it emits the following events.

Sr.No.	Event	Description
1	<b>exit</b>	Emitted when the process is about to exit. There is no way to prevent the exiting of the event loop at this point, and once all exit listeners have finished running the process will exit.
2	<b>beforeExit</b>	This event is emitted when node empties its event loop and has nothing else to schedule. Normally, node exits when there is no work scheduled, but a listener for 'beforeExit' can make asynchronous calls, and cause node to continue.

3	<b>uncaughtException</b>	Emitted when an exception bubbles all the way back to the event loop. If a listener is added for this exception, the default action <code>which is to print a stack trace and exit</code> will not occur.
4	<b>Signal Events</b>	Emitted when the processes receives a signal such as SIGINT, SIGHUP, etc.

## Properties

Process provides many useful properties to get better control over system interactions.

Sr.No.	Property	Description
1	<b>stdout</b>	A Writable Stream to stdout.
2	<b>stderr</b>	A Writable Stream to stderr.
3	<b>stdin</b>	A Writable Stream to stdin.
4	<b>argv</b>	An array containing the command line arguments. The first element will be 'node', the second element will be the name of the JavaScript file. The next elements will be any additional command line arguments.
5	<b>execPath</b>	This is the absolute pathname of the executable that started the process.

6	<b>execArgv</b>	This is the set of node-specific command line options from the executable that started the process.
7	<b>env</b>	An object containing the user environment.
8	<b>exitCode</b>	A number which will be the process exit code, when the process either exits gracefully, or is exited via process.exit without specifying a code.
9	<b>version</b>	A compiled-in property that exposes NODE_VERSION.
10	<b>versions</b>	A property exposing version strings of node and its dependencies.
11	<b>config</b>	An Object containing the JavaScript representation of the configure options that were used to compile the current node executable. This is the same as the "config.gypi" file that was produced when running the ./configure script.
12	<b>pid</b>	The PID of the process.
13	<b>title</b>	Getter/setter to set what is displayed in 'ps'.
14	<b>arch</b>	What processor architecture you're running on: 'arm', 'ia32', or 'x64'.

15	<b>platform</b>	What platform you're running on: 'darwin', 'freebsd', 'linux', 'sunos' or 'win32'
16	<b>mainModule</b>	Alternate way to retrieve require.main. The difference is that if the main module changes at runtime, require.main might still refer to the original main module in modules that were required before the change occurred. Generally it's safe to assume that the two refer to the same module.

## Methods

Process provides many useful methods to get better control over system interactions.

Sr.No.	Method	Description
1	<b>abort</b>	This causes node to emit an abort. This will cause node to exit and generate a core file.
2	<b>chdirdirectory</b>	Changes the current working directory of the process or throws an exception if that fails.
3	<b>cwd</b>	Returns the current working directory of the process.
4	<b>exit[code]</b>	Ends the process with the specified code. If omitted, exit uses the 'success' code 0.
5	<b>getgid</b>	Gets the group identity of the process. This is the numerical group id, not the group name. This function is only available on POSIX platforms <i>i.e.not Windows, Android.</i>

6	<b>setgidid</b>	Sets the group identity of the process. <i>Seesetgid(2.)</i> This accepts either a numerical ID or a groupname string. If a groupname is specified, this method blocks while resolving it to a numerical ID. This function is only available on POSIX platforms <i>i.e.notWindows,Android.</i>
7	<b>getuid</b>	Gets the user identity of the process. This is the numerical id, not the username. This function is only available on POSIX platforms <i>i.e.notWindows,Android.</i>
8	<b>setuidid</b>	Sets the user identity of the process. <i>Seesetgid(2.)</i> This accepts either a numerical ID or a username string. If a username is specified, this method blocks while resolving it to a numerical ID. This function is only available on POSIX platforms <i>i.e.notWindows,Android.</i>
9	<b>getgroups</b>	Returns an array with the supplementary group IDs. POSIX leaves it unspecified if the effective group ID is included but node.js ensures it always is. This function is only available on POSIX platforms <i>i.e.notWindows,Android.</i>
10	<b>setgroupsgroups</b>	Sets the supplementary group IDs. This is a privileged operation, meaning you need to be root or have the CAP_SETGID capability. This function is only available on POSIX platforms <i>i.e.notWindows,Android.</i>
11	<b>initgroupsuser,extragroup</b>	Reads /etc/group and initializes the group access list, using all groups of which the user is a member. This is a privileged operation, meaning you need to be root or have the CAP_SETGID capability. This function is only available on POSIX platforms <i>i.e.notWindows,Android.</i>
12	<b>killpid[,signal]</b>	Send a signal to a process. pid is the process id and signal is the string describing the signal to send. Signal names are strings like 'SIGINT' or 'SIGHUP'. If omitted, the signal will be 'SIGTERM'.
13	<b>memoryUsage</b>	Returns an object describing the memory usage of the Node process measured in bytes.
14	<b>nextTickcallback</b>	Once the current event loop turn runs to completion, call the callback function.
15	<b>umask[mask]</b>	Sets or reads the process's file mode creation mask. Child processes inherit the mask from the parent process. Returns the old mask if mask argument is given, otherwise returns the current mask.
16	<b>uptime</b>	Number of seconds Node has been running.
17	<b>hrtime</b>	Returns the current high-resolution real time in a [seconds, nanoseconds] tuple Array. It is relative to an arbitrary time in the past. It is not related to the time of

day and therefore not subject to clock drift. The primary use is for measuring performance between intervals.

## Example

Create a js file named test.js in **C:\>Nodejs\_WorkSpace**.

*File: test.js*

```
var util = require('util');

//printing to console
process.stdout.write("Hello World!" + "\n");

//reading passed parameter
process.argv.forEach(function(val, index, array) {
 console.log(index + ': ' + val);
});

//executable path
console.log(process.execPath);

//print the current directory
console.log('Current directory: ' + process.cwd());

//print the process version
console.log('Current version: ' + process.version);

//print the memory usage
console.log(util.inspect(process.memoryUsage()));
```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

Verify the Output.

```
Hello World!
0: node
1: C:\Nodejs_WorkSpace\test.js
2: one
3: 2
4: three
C:\Program Files\nodejs\node.exe
Current directory: C:\Nodejs_WorkSpace
Current version: v0.10.36
{ rss: 9314304, heapTotal: 3047296, heapUsed: 1460196 }
```

# NODE .- Ø \$ MODULE

**os** module is used for few basic operating-system related utility functions. os module can be imported using following syntax.

```
var os = require("os")
```

## Methods

Sr. No.	method	Description
1	<b>os.tmpdir</b>	Returns the operating system's default directory for temp files.
2	<b>os.endianness</b>	Returns the endianness of the CPU. Possible values are "BE" or "LE".
3	<b>os.hostname</b>	Returns the hostname of the operating system.
4	<b>os.type</b>	Returns the operating system name.
5	<b>os.platform</b>	Returns the operating system platform.
6	<b>os.arch</b>	Returns the operating system CPU architecture. Possible values are "x64", "arm" and "ia32".
7	<b>os.release</b>	Returns the operating system release.

8	<b>os.uptime</b>	Returns the system uptime in seconds.
9	<b>os.loadavg</b>	Returns an array containing the 1, 5, and 15 minute load averages.
10	<b>os.totalmem</b>	Returns the total amount of system memory in bytes.
11	<b>os.freemem</b>	Returns the amount of free system memory in bytes.
12	<b>os.cpus</b>	Returns an array of objects containing information about each CPU/core installed: model, speed <i>inMHz</i> , and times <i>an object containing the number of milliseconds the CPU/cores spent in user, nice, sys, idle, and iowait</i> .
13	<b>os.networkInterfaces</b>	Get a list of network interfaces.

## Properties

Sr. No.	property	Description
1	<b>os.EOL</b>	A constant defining the appropriate End-of-line marker for the operating system.

## Example

Create a js file named test.js in **C:\>Nodejs\_WorkSpace**.

*File: test.js*

```
var os = require("os");
```

```
//endianness
console.log('endianness : ' + os.endianness());
//type
console.log('type : ' + os.type());
//platform
console.log('platform : ' + os.platform());
//totalmem
console.log('total memory : ' + os.totalmem() + " bytes.");
//freemem
console.log('free memory : ' + os.freemem() + " bytes.");
```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

Verify the Output.

```
endianness : LE
type : Windows_NT
platform : win32
total memory : 1072152576 bytes.
free memory : 461508608 bytes.
```

## N O D E . - B A S T H M O D U L E

---

**path** module is used for handling and transforming file paths. path module can be imported using following syntax.

```
var path = require("path")
```

## Properties

Process provides many useful properties to get better control over system interactions.

Sr.No.	Property	Description
1	<b>path.sep</b>	The platform-specific file separator. '\\' or '/'.

2	<b>path.delimiter</b>	The platform-specific path delimiter, ; or ':'.
3	<b>path.posix</b>	Provide access to aforementioned path methods but always interact in a posix compatible way.
4	<b>path.win32</b>	Provide access to aforementioned path methods but always interact in a win32 compatible way.

## Methods

Sr. No.	method	Description
1	<b>path.normalizep</b>	Normalize a string path, taking care of '..' and '.' parts.
2	<b>path.join[path1][,path2][,...]</b>	Join all arguments together and normalize the resulting path.
3	<b>path.resolve[from...],to</b>	Resolves to to an absolute path.
4	<b>path.isAbsolutepath</b>	Determines whether path is an absolute path. An absolute path will always resolve to the same location, regardless of the working directory.
5	<b>path.relativefrom,to</b>	Solve the relative path from from to to.
6	<b>path.dirnamep</b>	Return the directory name of a path. Similar to the Unix dirname command.
7	<b>path.basenamep[,ext]</b>	Return the last portion of a path. Similar to the Unix basename command.
8	<b>path.splitextp</b>	Return the extension of the path, from the last '.' to end of string in the last portion of the path. If there is no '.'

		in the last portion of the path or the first character of it is '.', then it returns an empty string.
9	<b>path.parsepathString</b>	Returns an object from a path string.
10	<b>path.formatpathObject</b>	Returns a path string from an object, the opposite of path.parse above.

## Example

Create a js file named test.js in **C:\>Nodejs\_WorkSpace**.

*File: test.js*

```
var path = require("path");

//normalization
console.log('normalization : ' + path.normalize('/test/test1//2slashes/1slash/tab/..'));

//join
console.log('joint path : ' + path.join('/test', 'test1', '2slashes/1slash', 'tab', '..'));

//resolve
console.log('resolve : ' + path.resolve('test.js'));

//extName
console.log('ext name : ' + path.extname('test.js'));
```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

Verify the Output.

```
normalization : \test\test1\2slashes\1slash
joint path : \test\test1\2slashes\1slash
resolve : C:\Nodejs_WorkSpace\test.js
ext name : .js
```

## N O D E . - N E T     M O D U L E

---

**net** module is used to create both servers and clients. It provides an asynchronous network wrapper. net module can be imported using following syntax.

```
var net = require("net")
```

## Methods

Sr. No.	method	Description
1	<b>net.createServer[options][,connectionListener]</b>	Creates a new TCP server. The connectionListener argument is automatically set as a listener for the 'connection' event.
2	<b>net.connectoptions[,connectionListener]</b>	A factory method, which returns a new 'net.Socket' and connects to the supplied address and port.
3	<b>net.createConnectionoptions[,connectionListener]</b>	A factory method, which returns a new 'net.Socket' and connects to the supplied address and port.
4	<b>net.connectport[,host][,connectListener]</b>	Creates a TCP connection to port on host. If host is omitted, 'localhost' will be assumed. The connectListener parameter will be added as an listener for the 'connect' event. Is a factory method which returns a new 'net.Socket'.
5	<b>net.createConnectionport[,host][,connectListener]</b>	Creates a TCP connection to port on host. If host is omitted, 'localhost' will be assumed. The connectListener parameter will be added as an listener for the 'connect' event. Is a factory method which returns a new 'net.Socket'.
6	<b>net.connectpath[,connectListener]</b>	Creates unix socket connection to path. The connectListener parameter will be added as an listener for the 'connect' event. A factory method which returns a new 'net.Socket'.
7	<b>net.createConnectionpath[,connectListener]</b>	Creates unix socket connection to path. The connectListener parameter will be added as an listener for the 'connect' event. A factory method which returns a new 'net.Socket'.
8	<b>net.isIPinput</b>	Tests if input is an IP address. Returns 0 for invalid strings, returns 4 for IP version 4

### CONNECT FRAMEWORK

```

var connect = require('connect'); //connect framework
var http = require('http');

var app = connect();

function doFirst(request,response, next)
{
 console.log("Welcome to connect.....");
 next(); // handle next object also.....
}

function doSecond(request,response, next)
{
 console.log("Welcome to connect Second.....");
 next();
}

//stack it.....
app.use(doFirst); // handle user request called middleware....
app.use(doSecond);
http.createServer(app).listen(8888);
console.log("Connect server is running....");

```

		addresses, and returns 6 for IP version 6 addresses.
9	<b>net.isIPv4input</b>	Returns true if input is a version 4 IP address, otherwise returns false.
10	<b>net.isIPv6input</b>	Returns true if input is a version 6 IP address, otherwise returns false.

## Class:net.Server

This class is used to create a TCP or local server.

### Methods

Sr. No.	method	Description
1	<b>server.listenport[,host][,backlog][,callback]</b>	Begin accepting connections on the specified port and host. If the host is omitted, the server will accept connections directed to any IPv4 address <i>INADDRANY</i> . A port value of zero will assign a random port.
2	<b>server.listenpath[,callback]</b>	Start a local socket server listening for connections on the given path.
3	<b>server.listenhandle[,callback]</b>	The handle object can be set to either a server or socket <i>anythingwithanunderlyinghandlemember</i> , or a {fd: <n>} object. This will cause the server to accept connections on the specified handle, but it is presumed that the file descriptor or handle has already been bound to a port or domain socket. Listening on a file descriptor is not supported on Windows.
4	<b>server.listeneoptions[,callback]</b>	The port, host, and backlog properties of options, as well as the optional callback function, behave as they do on a call to <code>server.listenport,[host],[backlog],[callback]</code> . Alternatively, the path option can be used to specify a UNIX socket.
5	<b>server.close[callback]</b>	finally closed when all connections are ended and the server emits a 'close' event.

6	<b>server.address</b>	Returns the bound address, the address family name and port of the server as reported by the operating system.
7	<b>server.unref</b>	Calling unref on a server will allow the program to exit if this is the only active server in the event system. If the server is already unrefd calling unref again will have no effect.
8	<b>server.ref</b>	Opposite of unref, calling ref on a previously unrefd server will not let the program exit if it's the only server left <i>the default behavior</i> . If the server is refd calling ref again will have no effect.
9	<b>server.getConnectionscallback</b>	Asynchronously get the number of concurrent connections on the server. Works when sockets were sent to forks. Callback should take two arguments err and count.

## Events

Sr. No.	event	Description
1	<b>listening</b>	Emitted when the server has been bound after calling server.listen.
2	<b>connection</b>	Emitted when a new connection is made. Socket object, The connection object is available to event handler. Socket is an instance of net.Socket.
3	<b>close</b>	Emitted when the server closes. Note that if connections exist, this event is not emitted until all connections are ended.
4	<b>error</b>	Emitted when an error occurs. The 'close' event will be called directly following this event.

## Class:net.Socket

This object is an abstraction of a TCP or local socket. `net.Socket` instances implement a duplex Stream interface. They can be created by the user and used as a client `withconnect()` or they can be created by Node and passed to the user through the 'connection' event of a server.

## Events

`net.Socket` is an `eventEmitter` and it emits the following events.

Sr.No.	Event	Description
1	<b>lookup</b>	Emitted after resolving the hostname but before connecting. Not applicable to UNIX sockets.
2	<b>connect</b>	Emitted when a socket connection is successfully established.
3	<b>data</b>	Emitted when data is received. The argument <code>data</code> will be a Buffer or String. Encoding of data is set by <code>socket.setEncoding</code> .
4	<b>end</b>	Emitted when the other end of the socket sends a FIN packet.
5	<b>timeout</b>	Emitted if the socket times out from inactivity. This is only to notify that the socket has been idle. The user must manually close the connection.
6	<b>drain</b>	Emitted when the write buffer becomes empty. Can be used to throttle uploads.
7	<b>error</b>	Emitted when an error occurs. The 'close' event will be called directly following this event.
8	<b>close</b>	Emitted once the socket is fully closed. The argument <code>had_error</code> is a boolean which says if the socket was closed due to a transmission error.

## Properties

`net.Socket` provides many useful properties to get better control over socket interactions.

Sr.No.	Property	Description

1	<b>socket.bufferSize</b>	This property shows the number of characters currently buffered to be written.
2	<b>socket.remoteAddress</b>	The string representation of the remote IP address. For example, '74.125.127.100' or '2001:4860:a005::68'.
3	<b>socket.remoteFamily</b>	The string representation of the remote IP family. 'IPv4' or 'IPv6'.
4	<b>socket.remotePort</b>	The numeric representation of the remote port. For example, 80 or 21.
5	<b>socket.localAddress</b>	The string representation of the local IP address the remote client is connecting on. For example, if you are listening on '0.0.0.0' and the client connects on '192.168.1.1', the value would be '192.168.1.1'.
6	<b>socket.localPort</b>	The numeric representation of the local port. For example, 80 or 21.
7	<b>socket.bytesRead</b>	The amount of received bytes.
8	<b>socket.bytesWritten</b>	The amount of bytes sent.

## Methods

Sr. No.	method	Description
1	<b>new net.Socket[options]</b>	Construct a new socket object.
2	<b>socket.connectport[,host][,connectListener]</b>	Opens the connection for a given socket. If port and host are given, then the socket will be opened as a TCP socket, if host is omitted, localhost will be assumed. If a path is given, the socket will be opened as a unix socket to that path.
3	<b>socket.connectpath[,connectListener]</b>	Opens the connection for a given socket. If port and host are given, then the socket will be opened as a TCP socket, if host is omitted, localhost will be assumed. If a path is given, the socket will be opened as a unix socket to that path.
4	<b>socket.setEncoding[encoding]</b>	Set the encoding for the socket as a Readable Stream.
5	<b>socket.write(data[,encoding][,callback])</b>	Sends data on the socket. The second parameter specifies the encoding in the case of a string--it defaults to UTF8 encoding.
6	<b>socket.end[data][,encoding]</b>	Half-closes the socket. i.e., it sends a FIN packet. It is possible the server will still send some data.
7	<b>socket.destroy</b>	Ensures that no more I/O activity happens on this socket. Only necessary in case of errors <i>parseerror</i> or <i>so</i> .
8	<b>socket.pause</b>	Pauses the reading of data. That is, 'data' events will not be emitted. Useful to throttle back an upload.
9	<b>socket.resume</b>	Resumes reading after a call to pause.
10	<b>socket.setTimeout(timeout[,callback])</b>	Sets the socket to timeout after timeout milliseconds of inactivity on the socket. By default net.Socket do not have a timeout.

11	<b>socket.setNoDelay[noDelay]</b>	Disables the Nagle algorithm. By default TCP connections use the Nagle algorithm, they buffer data before sending it off. Setting true for noDelay will immediately fire off data each time socket.write is called. noDelay defaults to true.
12	<b>socket.setKeepAlive[enable][,initialDelay]</b>	Enable/disable keep-alive functionality, and optionally set the initial delay before the first keepalive probe is sent on an idle socket. enable defaults to false.
13	<b>socket.address</b>	Returns the bound address, the address family name and port of the socket as reported by the operating system. Returns an object with three properties, e.g. { port: 12346, family: 'IPv4', address: '127.0.0.1' }.
14	<b>socket.unref</b>	Calling unref on a socket will allow the program to exit if this is the only active socket in the event system. If the socket is already unrefed calling unref again will have no effect.
15	<b>socket.ref</b>	Opposite of unref, calling ref on a previously unrefed socket will not let the program exit if it's the only socket left <i>the default behavior</i> . If the socket is refd calling ref again will have no effect.

## Example

Create a js file named server.js in **C:\>Nodejs\_WorkSpace**.

*File: server.js*

```
var net = require('net');

var server = net.createServer(function(connection) {
 console.log('client connected');
 connection.on('end', function() {
 console.log('client disconnected');
 });
 connection.write('Hello World!\r\n');
 connection.pipe(connection);
});

server.listen(8080, function() {
 console.log('server is listening');
});
```

Now run the server.js to see the result:

```
C:\Nodejs_WorkSpace>node server.js
```

Verify the Output.

```
server is listening
```

Create a js file named client.js in C:\>Nodejs\_WorkSpace.

*File: client.js*

```
var net = require('net');

var client = net.connect({port: 8080}, function() {
 console.log('connected to server!');
});

client.on('data', function(data) {
 console.log(data.toString());
 client.end();
});

client.on('end', function() {
 console.log('disconnected from server');
});
```

Now run the client.js in another terminal to see the result:

```
C:\Nodejs_WorkSpace>node client.js
```

Verify the Output.

```
connected to server!
Hello World!

disconnected from server
```

Verify the Output on terminal where server.js is running.

```
server is listening
client connected
client disconnected
```

## N O D E . - D N S M O D U L E

**dns** module is used to do actual DNS lookup as well as to use underlying operating system name resolution functionalities.. It provides an asynchronous network wrapper. dns module can be imported using following syntax.

```
var dns = require("dns")
```

## Methods

Sr. No.	method	Description
1	<b>dns.lookuphostname[,options],callback</b>	Resolves a hostname <i>e.g. 'google.com'</i> into the first found A IPv4 or AAAA records. options can be an object or integer. If options is not provided, then IP v4 and v6 addresses are both valid. If options is an integer, then it must be 4 or 6.
2	<b>dns.lookupServiceaddress,port,callback</b>	Resolves the given address and port into a hostname and service using generic queries.
3	<b>dns.resolvehostname[,rrtype],callback</b>	Resolves a hostname <i>e.g. 'google.com'</i> into an array of the record types specified by rrtype.
4	<b>dns.resolve4hostname,callback</b>	The same as dns.resolve, but only for IPv4 queries <i>Arecords</i> . addresses is an array of IPv4 addresses <i>e.g. ['74.125.79.104','74.125.79.105','74.125.79.106']</i> .
5	<b>dns.resolve6hostname,callback</b>	The same as dns.resolve4 except for IPv6 queries <i>anAAAAquery</i> .
6	<b>dns.resolveMxhostname,callback</b>	The same as dns.resolve, but only for mail exchange queries <i>MXrecords</i> .
7	<b>dns.resolveTxthostname,callback</b>	The same as dns.resolve, but only for text queries <i>TXTrecords</i> . addresses is an array of the text records available for hostname <i>e.g.,[['v=spf1ip4:0.0.0.0','all']]</i> . array contains TXT chunks of one record. Depending on the use case, the chunks can either joined together or treated separately.
8	<b>dns.resolveSrvhostname,callback</b>	The same as dns.resolve, but only for service records <i>SRVrecords</i> . addresses is an array of the SRV records available for hostname. Properties of SRV records are priority, weight, port, and name <i>e.g.,['priority':10,'weight':5,'port':21223,'name':'service.example.com']</i> .
9	<b>dns.resolveSoahostname,callback</b>	The same as dns.resolve, but only for start of authority record queries <i>SODNSrecords</i> .
10	<b>dns.resolveNshostname,callback</b>	The same as dns.resolve, but only for name server records <i>NSrecords</i> . addresses is an array of the name server records available for hostname <i>e.g.,['ns1.example.com','ns2.example.com']</i> .
11	<b>dns.resolveCnamehostname,callback</b>	The same as dns.resolve, but only for canonical name records <i>CNAMErecords</i> . addresses is an array of the canonical name records available for hostname <i>e.g.,['bar.example.com']</i> .
12	<b>dns.reverseip,callback</b>	Reverse resolves an ip address to an array of hostnames.
13	<b>dns.getServers</b>	Returns an array of IP addresses as strings that are currently being used for DNS resolution.
14	<b>dns.setServersservers</b>	Given an array of IP addresses as strings, set them as the servers to use for DNS resolution.

## rrtypes

Following is the list of valid rrtypes used by dns.resolve method

- **A** - IPV4 addresses, default
- **AAAA** - IPV6 addresses
- **MX** - mail exchange records
- **TXT** - text records
- **SRV** - SRV records
- **PTR** - used for reverse IP lookups
- **NS** - name server records
- **CNAME** - canonical name records
- **SOA** - start of authority record

## Error Codes

Each DNS query can return one of the following error codes:

- **dns.NODATA** - DNS server returned answer with no data.
- **dns.FORMERR** - DNS server claims query was misformatted.
- **dns.SERVFAIL** - DNS server returned general failure.
- **dns.NOTFOUND** - Domain name not found.
- **dns.NOTIMP** - DNS server does not implement requested operation.
- **dns.REFUSED** - DNS server refused query.
- **dns.BADQUERY** - Misformatted DNS query.
- **dns.BADNAME** - Misformatted hostname.
- **dns.BADFAMILY** - Unsupported address family.
- **dns.BADRESP** - Misformatted DNS reply.
- **dns.CONNREFUSED** - Could not contact DNS servers.
- **dns.TIMEOUT** - Timeout while contacting DNS servers.
- **dns.EOF** - End of file.
- **dns.FILE** - Error reading file.
- **dns.NOMEM** - Out of memory.
- **dns.DESTRUCTURE** - Channel is being destroyed.
- **dns.BADSTR** - Misformatted string.
- **dns.BADFLAGS** - Illegal flags specified.
- **dns.NONAME** - Given hostname is not numeric.
- **dns.BADHINTS** - Illegal hints flags specified.
- **dns.NOTINITIALIZED** - c-ares library initialization not yet performed.
- **dns.LOADIPHLPAPI** - Error loading iphlpapi.dll.
- **dns.ADDRGETNETWORKPARAMS** - Could not find GetNetworkParams function.
- **dns.CANCELLED** - DNS query cancelled.

## Example

Create a js file named test.js in **C:\>Nodejs\_WorkSpace**.

*File: test.js*

```
var dns = require('dns');

dns.lookup('www.google.com', function onLookup(err, address, family) {
 console.log('address:', address);
```

```

dns.reverse(address, function (err, hostnames) {
 if (err) {
 console.log(err.stack);
 }

 console.log('reverse for ' + address + ': ' + JSON.stringify(hostnames));
});
});

```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

Verify the Output.

```
address: 74.125.200.103
reverse for 74.125.200.103: ["sa-in-f103.1e100.net"]
```

## N O D E . - D O M A I N   M O D U L E

---

**domain** module is used to intercept unhandled error. These unhandled error can be intercepted using internal binding or external binding. If errors are not handled at all then they will simply crash the Node application.

- **Internal Binding** - Error emitter is executing its code within run method of a domain.
- **External Binding** - Error emitter is added explicitly to a domain using its add method.

domain module can be imported using following syntax.

```
var domain = require("domain")
```

Domain class of domain module is used to provide functionality of routing errors and uncaught exceptions to the active Domain object. It is a child class of EventEmitter. To handle the errors that it catches, listen to its error event. It is created using following syntax:

```
var domain = require("domain");
var domain1 = domain.create();
```

## Methods

Sr. No.	method	Description

1	<b>domain.runfunction</b>	Run the supplied function in the context of the domain, implicitly binding all event emitters, timers, and lowlevel requests that are created in that context. This is the most basic way to use a domain.
2	<b>domain.addemitter</b>	Explicitly adds an emitter to the domain. If any event handlers called by the emitter throw an error, or if the emitter emits an error event, it will be routed to the domain's error event, just like with implicit binding.
3	<b>domain.removeemitter</b>	The opposite of <code>domain.addemitter</code> . Removes domain handling from the specified emitter.
4	<b>domain.bindcallback</b>	The returned function will be a wrapper around the supplied callback function. When the returned function is called, any errors that are thrown will be routed to the domain's error event.
5	<b>domain.interceptcallback</b>	This method is almost identical to <code>domain.bindcallback</code> . However, in addition to catching thrown errors, it will also intercept Error objects sent as the first argument to the function.
6	<b>domain.enter</b>	The enter method is plumbing used by the run, bind, and intercept methods to set the active domain. It sets <code>domain.active</code> and <code>process.domain</code> to the domain, and implicitly pushes the domain onto the domain stack managed by the domain module <code>seedomain.exit</code> (for details on the domain stack). The call to enter delimits the beginning of a chain of asynchronous calls and I/O operations bound to a domain.
7	<b>domain.exit</b>	The exit method exits the current domain, popping it off the domain stack. Any time execution is going to switch to the context of a different chain of asynchronous calls, it's important to ensure that the current domain is exited. The call to exit delimits either the end of or an interruption to the chain of asynchronous calls and I/O operations bound to a domain.
8	<b>domain.dispose</b>	Once dispose has been called, the domain will no longer be used by callbacks bound into the domain via run, bind, or intercept, and a dispose event is emit

## Properties

Sr.No.	Property	Description

1	<b>domain.members</b>	An array of timers and event emitters that have been explicitly added to the domain.
---	-----------------------	--------------------------------------------------------------------------------------

## Example

Create a js file named test.js in **C:\>Nodejs\_WorkSpace**.

*File: test.js*

```
var EventEmitter = require("events").EventEmitter;
var domain = require("domain");

var emitter1 = new EventEmitter();

//Create a domain
var domain1 = domain.create();

domain1.on('error', function(err) {
 console.log("domain1 handled this error ("+err.message+")");
});

//explicit binding
domain1.add(emitter1);

emitter1.on('error',function(err) {
 console.log("listener handled this error ("+err.message+")");
});

emitter1.emit('error',new Error('To be handled by listener'));

emitter1.removeAllListeners('error');

emitter1.emit('error',new Error('To be handled by domain1'));

var domain2 = domain.create();

domain2.on('error', function(err) {
 console.log("domain2 handled this error ("+err.message+")");
});
```

```
//implicit binding

domain2.run(function() {
 var emitter2 = new EventEmitter();

 emitter2.emit('error',new Error('To be handled by domain2'));
}) ;

domain1.remove(emitter1);

emitter1.emit('error',new Error('Converted to exception. System will crash!'));
```

Now run the test.js to see the result:

```
C:\Nodejs_WorkSpace>node test.js
```

### Verify the Output.

```
listener handled this error (To be handled by listener)
domain1 handled this error (To be handled by domain1)
domain2 handled this error (To be handled by domain2)

events.js:72
 throw er; // Unhandled 'error' event
 ^
Error: Converted to exception. System will crash!
 at Object.<anonymous> (C:\Nodejs_WorkSpace\test.js:42:23)
 at Module._compile (module.js:456:26)
 at Object.Module._extensions..js (module.js:474:10)
 at Module.load (module.js:356:32)
 at Function.Module._load (module.js:312:12)
 at Function.Module.runMain (module.js:497:10)
 at startup (node.js:119:16)
 at node.js:929:3
```

## N O D E . - W I S B   M O D U L E

---

### Introduction to Web Server

Web Server is a software application which processes request using HTTP protocol and returns web pages as response to the clients. Web servers usually delivers html documents along with images, style sheets and scripts. Most web server also support server side scripts using scripting language or redirect to application server which perform the specific task of getting data from database, perform complex logic etc. Web server then returns the output of the application server to client.

Apache web server is one of the most common web server being used. It is an open source project.

### Path locating process

Web server maps the path of a file using URL, Uniform Resource Locator. It can be a local file system or a external/internal program. For example:

A client makes a request using browser, URL: <http://www.test-example-site.com/website/index.htm>.

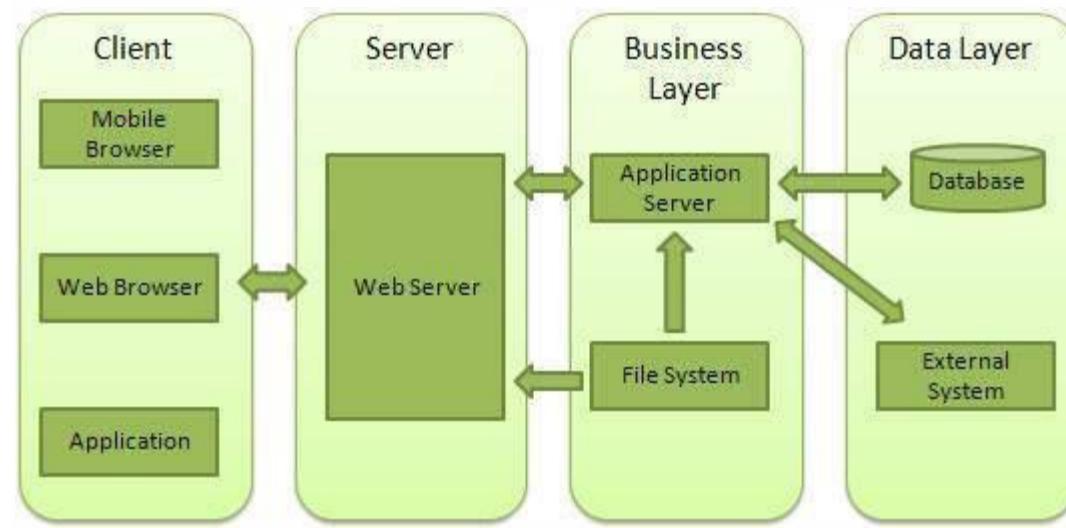
Browser will make request as:

```
GET /website/index.htm HTTP /1.1
HOST www.test-example-site.com
```

Web Server will append the path to its root directory. Consider, for example the root directory is home/www then actual path will be translated to home/www/website/index.htm.

## Introduction to web architecture

Web application are using divided into four layers:



- **Client** - This layer consists of web browsers, mobile browsers or applications which can make HTTP request to server.
- **Server** - This layer consists of Web server which can intercepts the request made by clients and pass them the response.
- **Business** - This layer consists of application server which is utilized by web server to do dynamic tasks. This layer interacts with data layer via data base or some external programs.
- **Data** - This layer consists of databases or any source of data.

## Creating Web Server using Node

Create an HTTP server using `http.createServer` method. Pass it a function with parameters `request` and `response`. Write the sample implementation to return a requested page. Pass a port 8081 to `listen` method.

Create a js file named `server.js` in **C:\>Nodejs\_WorkSpace**.

*File: server.js*

```
//http module is required to create a web server
var http = require('http');
//fs module is required to read file from file system
var fs = require('fs');
//url module is required to parse the URL passed to server
```

```

var url = require('url');

//create the server

http.createServer(function (request, response) {
 //parse the pathname containing file name
 var pathname = url.parse(request.url).pathname;
 //print the name of the file for which request is made.
 //if url is http://localhost:8081/test.htm then
 //pathname will be /test.htm
 console.log("Request for " + pathname + " received.");
 //read the requested file content from file system
 fs.readFile(pathname.substr(1), function (err, data) {
 //if error occurred during file read
 //send an error response to client
 //that web page is not found.
 if (err) {
 console.log(err.stack);
 // HTTP Status: 404 : NOT FOUND
 // Content Type: text/plain
 response.writeHead(404, {'Content-Type': 'text/html'});
 } else{
 //Page found
 // HTTP Status: 200 : OK
 // Content Type: text/plain
 response.writeHead(200, {'Content-Type': 'text/html'});
 // write the content of the file to response body
 response.write(data.toString());
 }
 // send the response body
 response.end();
 });
}).listen(8081);
// console will print the message
console.log('Server running at http://127.0.0.1:8081/');

```

Create a htm file named test.htm in **C:\>Nodejs\_WorkSpace**.

*File: test.htm*

```

<html>
<head>
<title>Sample Page</title>

```

```
</head>
<body>
Hello World!
</body>
</html>
```

Now run the server.js to see the result:

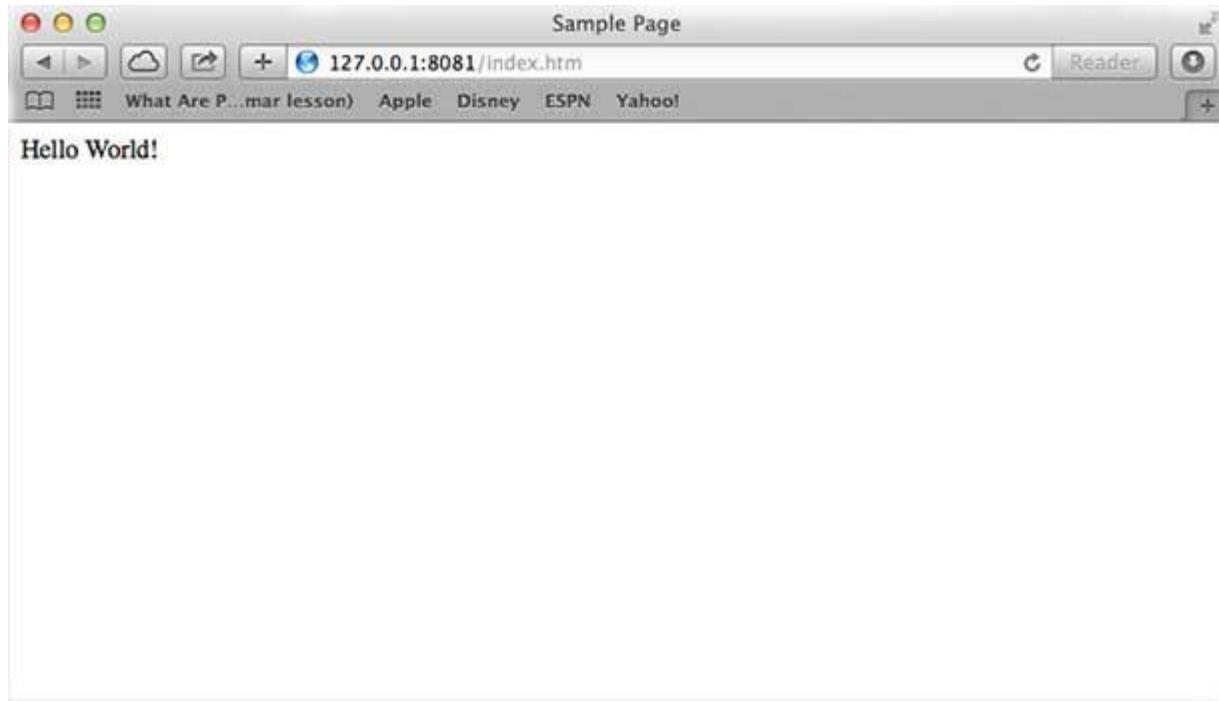
```
C:\Nodejs_WorkSpace>node server.js
```

Verify the Output. Server has started

```
Server running at http://127.0.0.1:8081/
```

## Make a request to Node.js server

Open <http://127.0.0.1:8081/test.htm> in any browser and see the below result.



Verify the Output at server end.

```
Server running at http://127.0.0.1:8081/
Request for /test.htm received.
```

## Creating Web client using Node

A web client can be created using http module. See the below example:

Create a js file named client.js in C:\>Nodejs\_WorkSpace.

*File: client.js*

```

//http module is required to create a web client
var http = require('http');

//options are to be used by request
var options = {
 host: 'localhost',
 port: '8081',
 path: '/test.htm'
};

//callback function is used to deal with response
var callback = function(response) {
 // Continuously update stream with data
 var body = '';
 response.on('data', function(data) {
 body += data;
 });
 response.on('end', function() {
 // Data received completely.
 console.log(body);
 });
}

//make a request to the server
var req = http.request(options, callback);
req.end();

```

Now run the client.js in a different command terminal other than of server.js to see the result:

```
C:\Nodejs_WorkSpace>node client.js
```

**Verify the Output.**

```

<html>
<head>
<title>Sample Page</title>
</head>
<body>
Hello World!
</body>
</html>

```

**Verify the Output at server end.**

```
Server running at http://127.0.0.1:8081/
Request for /test.htm received.
```

## Express Overview

Express JS is a very popular web application framework built to create Node JS Web based applications. It provides an integrated environment to facilitate rapid development of Node based Web applications. Express framework is based on Connect middleware engine and used Jade html template framework for HTML templating. Following are some of the core features of Express framework:

- Allows to set up middlewares to respond to HTTP Requests.
- Defines a routing table which is used to perform different action based on HTTP Method and URL.
- Allows to dynamically render HTML Pages based on passing arguments to templates.

## Installing Express

Firstly, install the Express framework globally using npm so that it can be used to create web application using node terminal.

```
C:\Nodejs_WorkSpace>npm install express -g
```

Once npm completes the download, you can verify by looking at the content of <user-directory>/npm/node\_modules. Or type the following command:

```
C:\Nodejs_WorkSpace>npm ls -g
```

You will see the following output:

```
C:\Documents and Settings\Administrator\Application Data\npm
+-- express@4.11.2
 +-- accepts@1.2.3
 | +-- mime-types@2.0.8
 | | +-- mime-db@1.6.1
 | +-- negotiator@0.5.0
 +-- content-disposition@0.5.0
 +-- cookie@0.1.2
 +-- cookie-signature@1.0.5
 +-- debug@2.1.1
 | +-- ms@0.6.2
 +-- depd@1.0.0
 +-- escape-html@1.0.1
 +-- etag@1.5.1
 | +-- crc@3.2.1
 +-- finalhandler@0.3.3
 +-- fresh@0.2.4
 +-- media-typer@0.3.0
 +-- merge-descriptors@0.0.2
 +-- methods@1.1.1
 +-- on-finished@2.2.0
 | +-- ee-first@1.1.0
 +-- parseurl@1.3.0
 +-- path-to-regexp@0.1.3
 +-- proxy-addr@1.0.6
 | +-- forwarded@0.1.0
 | +-- ipaddr.js@0.1.8
 +-- qs@2.3.3
```

```
+-+ range-parser@1.0.2
+-+ send@0.11.1
| +-+ destroy@1.0.3
| +-+ mime@1.2.11
| +-+ ms@0.7.0
+-+ serve-static@1.8.1
+-+ type-is@1.5.6
| +-+ mime-types@2.0.8
| +-+ mime-db@1.6.1
+-+ utils-merge@1.0.0
+-+ vary@1.0.0
```

## Express Generator

Now install the express generator using npm. Express generator is used to create an application skeleton using express command.

```
C:\Nodejs_WorkSpace> npm install express-generator -g
```

You will see the following output:

```
C:\Nodejs_WorkSpace>npm install express-generator -g
C:\Documents and Settings\Administrator\Application Data\npm\express -> C:\Documents and Settings\Administrator\Application Data\npm\node_modules\express-generator\bin\express
express-generator@4.12.0 C:\Documents and Settings\Administrator\Application Data\npm\node_modules\express-generator
+-+ sorted-object@1.0.0
+-+ commander@2.6.0
+-+ mkdirp@0.5.0 (minimist@0.0.8)
```

## Hello world Example

Now create a sample application say firstApplication using the following command:

```
C:\Nodejs_WorkSpace> express firstApplication
```

You will see the following output:

```
create : firstApplication
create : firstApplication/package.json
create : firstApplication/app.js
create : firstApplication/public
create : firstApplication/public/javascripts
create : firstApplication/public/images
create : firstApplication/public/stylesheets
create : firstApplication/public/stylesheets/style.css
create : firstApplication/routes
create : firstApplication/routes/index.js
create : firstApplication/routes/users.js
create : firstApplication/views
create : firstApplication/views/index.jade
create : firstApplication/views/layout.jade
create : firstApplication/views/error.jade
create : firstApplication/bin
create : firstApplication/bin/www

install dependencies:
$ cd firstApplication && npm install
```

```
run the app:
$ DEBUG=firstApplication:* ./bin/www
```

Move to firstApplication folder and install dependencies of firstApplication using the following command:

```
C:\Nodejs_WorkSpace\firstApplication> npm install
```

You will see the following output:

```
debug@2.1.2 node_modules\debug
+-- ms@0.7.0

cookie-parser@1.3.4 node_modules\cookie-parser
+-- cookie-signature@1.0.6
+-- cookie@0.1.2

morgan@1.5.1 node_modules\morgan
+-- basic-auth@1.0.0
+-- depd@1.0.0
+-- on-finished@2.2.0 (ee-first@1.1.0)

serve-favicon@2.2.0 node_modules\serve-favicon
+-- ms@0.7.0
+-- fresh@0.2.4
+-- parseurl@1.3.0
+-- etag@1.5.1 (crc@3.2.1)

jade@1.9.2 node_modules\jade
+-- character-parser@1.2.1
+-- void-elements@2.0.1
+-- commander@2.6.0
+-- mkdirp@0.5.0 (minimist@0.0.8)
+-- transformers@2.1.0 (promise@2.0.0, css@1.0.8, uglify-js@2.2.5)
+-- with@4.0.1 (acorn-globals@1.0.2, acorn@0.11.0)
+-- constantinople@3.0.1 (acorn-globals@1.0.2)

express@4.12.2 node_modules\express
+-- merge-descriptors@1.0.0
+-- cookie-signature@1.0.6
+-- methods@1.1.1
+-- cookie@0.1.2
+-- fresh@0.2.4
+-- utils-merge@1.0.0
+-- range-parser@1.0.2
+-- escape-html@1.0.1
+-- parseurl@1.3.0
+-- vary@1.0.0
+-- content-type@1.0.1
+-- finalhandler@0.3.3
+-- serve-static@1.9.1
+-- content-disposition@0.5.0
+-- path-to-regexp@0.1.3
+-- depd@1.0.0
+-- qs@2.3.3
+-- on-finished@2.2.0 (ee-first@1.1.0)
+-- etag@1.5.1 (crc@3.2.1)
+-- proxy-addr@1.0.6 (forwarded@0.1.0, ipaddr.js@0.1.8)
+-- send@0.12.1 (destroy@1.0.3, ms@0.7.0, mime@1.3.4)
+-- accepts@1.2.4 (negotiator@0.5.1, mime-types@2.0.9)
+-- type-is@1.6.0 (media-typewriter@0.3.0, mime-types@2.0.9)

body-parser@1.12.0 node_modules\body-parser
+-- content-type@1.0.1
+-- bytes@1.0.0
```

```
+-- raw-body@1.3.3
+-- depd@1.0.0
+-- qs@2.3.3
+-- iconv-lite@0.4.7
+-- on-finished@2.2.0 (ee-first@1.1.0)
+-- type-is@1.6.0 (media-type@0.3.0, mime-types@2.0.9)
```

Here express generator has created a complete application structure which you can verify as firstApplication folder gets created in Nodejs\_WorkSpace folder with following folders/files:

```
.
+-- app.js
+-- bin
| +-- www
+-- package.json
+-- public
| +-- images
| +-- javascripts
| +-- stylesheets
| +-- style.css
+-- routes
| +-- index.js
| +-- users.js
+-- views
 +-- error.jade
 +-- index.jade
 +-- layout.jade
```

- **package.json** Application descriptor file contains dependencies list and other attributes of the application which Node utilizes.
- **app.js** Contains initialization code for server.
- **bin** Used to store the application in production mode.
- **public** Used to store the images, stylesheets and javascript files
- **routes** Contains route handlers
- **views** Contains html templates to generate various views for web application.

## First Application

app.js is the core engine of express based application. Let's update the default app.js to include port information and creates a server using it. Add the following lines to app.js

```
//set the server port
app.set('port', process.env.PORT || 3000);

//create the server
http.createServer(app).listen(app.get('port'), function() {
 console.log('Express server listening on port ' + app.get('port'));
});
```

## Updated app.js

Following are the full content of the app.js file

Update app.js file present in **C:\>Nodejs\_WorkSpace\firstApplication**.

## File: app.js

```
var express = require('express');
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
var http = require('http');

var routes = require('./routes/index');
var users = require('./routes/users');

var app = express();

// view engine setup
app.set('port', process.env.PORT || 3000);
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade'); console.log(__dirname);
console.log(__filename);

display directory name and file name

// uncomment after placing your favicon in /public
//app.use(favicon(__dirname + '/public/favicon.ico'));
app.use(logger('dev')); hide this error messages when it goes to production.
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', routes);
app.use('/users', users);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
 var err = new Error('Not Found');
 err.status = 404;
 next(err);
});

// error handlers

// development error handler
```

```

// will print stacktrace
if (app.get('env') === 'development') {
 app.use(function(err, req, res, next) {
 res.status(err.status || 500);
 res.render('error', {
 message: err.message,
 error: err
 });
 });
}

// production error handler
// no stacktraces leaked to user
app.use(function(err, req, res, next) {
 res.status(err.status || 500);
 res.render('error', {
 message: err.message,
 error: {}
 });
});

http.createServer(app).listen(app.get('port'), function() {
 console.log('Express server listening on port ' + app.get('port'));
});

module.exports = app;

```

during development it displays error messages.....

Now run the app.js to see the result:

```
C:\Nodejs_WorkSpace\firstApplication>node app
```

Verify the Output. Server has started

```
Express server listening on port 3000
```

## Make a request to firstApplication

Open <http://localhost:3000/> in any browser and see the below result.



## Basic Routing

Following code in app.js binds two route handlers.

```
var routes = require('./routes/index');
var users = require('./routes/users');

...
app.use('/', routes);
app.use('/users', users);
```

- **routes** - routes *index.js*, route handler handles all request made to home page via localhost:3000
- **users** - users *users.js*, route handler handles all request made to /users via localhost:3000/users

Following is the code of **C:\>Nodejs\_WorkSpace\firstApplication\routes\index.js** created by express generator.

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
 res.render('index', { title: 'Express' });
});

module.exports = router;
```

When node server gets a request for home page, express router render the index page using **index.jade** template while passing a parameter **title** with value 'Express'. Following are the contents of **C:\>Nodejs\_WorkSpace\firstApplication\views\index.jade** template.

```
extends layout
```

```
block content
 h1= title
 p Welcome to #{title}
```

# N O D E . - E N H A N C I N G F I I R C S A T T I A O P N P L

## Overview

In this article, we'll enhance the express js application created in [Express Application](#) chapter to do the following functionalities:

- Show list of all the users.
- Show details of a particular user.
- Add details of new user.

## Step 1: Create a JSON based database

Firstly, let's create a sample json based database of users.

Create a json file named user.json in **C:\>Nodejs\_WorkSpace\firstApplication**.

*File: user.json*

```
{
 "user1" : {
 "name" : "mahesh",
 "password" : "password1",
 "profession" : "teacher"
 },
 "user2" : {
 "name" : "suresh",
 "password" : "password2",
 "profession" : "librarian"
 },
 "user3" : {
 "name" : "ramesh",
 "password" : "password3",
 "profession" : "clerk"
 }
}

//app.locals.points = "8000"; // it makes the variable local to APP
app.locals.JSONFile = require('./JSONFile.json');
```

## Step 2: Create Users specific Jade Views

Create a **user** directory in **C:\>Nodejs\_WorkSpace\firstApplication\views** directory with the following views.

- **index.jade** - View to show list of all users.
- **new.jade** - View to show a form to add a new user.
- **profile.jade** - View to show detail of an user

Create a index.jade in **C:\>Nodejs\_WorkSpace\firstApplication\views\users**.

*File: index.jade*

```
h1 Users

p
 a(href="/users/new/") Create new user
ul
 - for (var username in users) {
 li
 a(href="/users/" + encodeURIComponent(username))= users[username].name
 - }

```

Create a new.jade in **C:\>Nodejs\_WorkSpace\firstApplication\views\users**.

*File: index.jade*

```
h1 New User

form(method="POST" action="/Users/addUser")
 p
 label(for="name") Name
 input#name(name="name")
 p
 label(for="password") Password
 input#name(name="password")
 p
 label(for="profession") Profession
 input#name(name="profession")
 p
 input(type="submit", value="Create")
```

Create a profile.jade in **C:\>Nodejs\_WorkSpace\firstApplication\views\users**.

*File: profile.jade*

```
h1 Name: #{user.name}
```

```
h2 Profession: #{user.profession}
```

### Step 3: Update users route handler, users.js

Update users.js in C:\>Nodejs\_WorksSpace\firstApplication\routes.

*File: users.js*

```
var express = require('express');
var router = express.Router();

var users = require('../users.json');

/* GET users listing. */
router.get('/', function(req, res) {
 res.render('users/index', { title: 'Users', users:users });
});

/* Get form to add a new user*/
router.get('/new', function(req, res) {
 res.render('users/new', { title: 'New User'});
});

/* Get detail of a new user */
router.get('/:name', function(req, res, next) {
 var user = users[req.params.name]
 if(user){
 res.render('users/profile', { title: 'User Profile', user:user});
 }else{
 next();
 }
});

/* post the form to add new user */
router.post('/addUser', function(req, res, next) {
 if(users[req.body.name]){
 res.send('Conflict', 409);
 }else{
 users[req.body.name] = req.body;
 res.redirect('/users/');
 }
});
```

```
module.exports = router;
```

Now run the app.js to see the result:

```
C:\Nodejs_WorkSpace\firstApplication>node app
```

Verify the Output. Server has started

```
Express server listening on port 3000
```

Make a request to firstApplication to get list of all the users. Open <http://localhost:3000/users> in any browser and see the below result.



Click on Create new User link to see a form.



Submit form and see the updated list.



Click on newly created user to see the details.

A screenshot of a web browser window showing user details. The address bar at the top shows the URL "localhost:3000/users/naresh". The main content area displays two lines of text: "Name: naresh" and "Profession: clerk".

You can check the server status also as following:

```
C:\Nodejs_WorkSpace\firstApplication>node app
Express server listening on port 3000
GET /users/ 200 809.161 ms - 201
GET /users/new/ 304 101.627 ms - -
GET /users/new/ 304 33.496 ms - -
POST /Users/addUser 302 56.206 ms - 70
GET /users/ 200 43.548 ms - 245
GET /users/naresh 200 12.313 ms - 47
```

## What is REST architecture?

REST stands for REpresentational State Transfer. REST is web standards based architecture and uses HTTP Protocol. It revolves around resource where every component is a resource and a resource is accessed by a common interface using HTTP standard methods. REST was first introduced by Roy Fielding in 2000.

In REST architecture, a REST Server simply provides access to resources and REST client accesses and modifies the resources. Here each resource is identified by URIs/ global IDs. REST uses various representation to represent a resource like text, JSON, XML. JSON is the most popular one.

## HTTP methods

Following four HTTP methods are commonly used in REST based architecture.

- **GET** - Provides a read only access to a resource.
- **PUT** - Used to create a new resource.
- **DELETE** - Used to remove a resource.
- **POST** - Used to update a existing resource or create a new resource.

## Introduction to RESTful web services

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability *e.g., between Java and Python, or Windows and Linux applications* is due to the use of open standards.

Web services based on REST Architecture are known as RESTful web services. These webservices uses HTTP methods to implement the concept of REST architecture. A RESTful web service usually defines a URI, Uniform Resource Identifier a service, provides resource representation such as JSON and set of HTTP Methods.

## Creating RESTful for A Library

Now, we'll enhance the express js application created in [Express Sample Application](#) chapter to create a webservice say user management with following functionalities:

Sr. No.	URI	HTTP Method	POST body	Result
1	/users/	GET	empty	Show list of all the users.

2	/users/addUser	POST	JSON String	Add details of new user.
3	/users/:id	GET	empty	Show details of a user.

## Getting All users

Firstly, let's update a sample json based database of users.

Update json file named user.json in **C:\>Nodejs\_WorkSpace\firstApplication**.

*File: user.json*

```
{
 "user1" : {
 "name" : "mahesh",
 "password" : "password1",
 "profession" : "teacher",
 "id": 1
 },
 "user2" : {
 "name" : "suresh",
 "password" : "password2",
 "profession" : "librarian",
 "id": 2
 },
 "user3" : {
 "name" : "ramesh",
 "password" : "password3",
 "profession" : "clerk",
 "id": 3
 }
}
```

When a client send a GET request to /users, server should send a response containing all the user. Update users route handler, users.js

Update users.js in **C:\>Nodejs\_WorkSpace\firstApplication\routes**.

*File: users.js*

```

/* GET users listing. */

router.get('/', function(req, res) {
 res.send({ title: 'Users', users:users });
}) ;

```

## Add details of new user.

When a client send a POST request to /users/addUser with body containing the JSON String, server should send a response stating the status. Update users route handler, users.js

Update users.js in **C:\>Nodejs\_WorkSpace\firstApplication\routes**.

*File: users.js*

```

/*add a user*/
router.post('/addUser', function(req, res, next) {
 var body = '';
 req.on('data', function (data) {
 body += data;
 });
 req.on('end', function () {
 var json = JSON.parse(body);
 users["user"+json.id] = body;
 res.send({ Message: 'User Added'});
 });
}) ;

```

## Show details of new user.

When a client send a GET request to /users with an id, server should send a response containing detail of that user. Update users route handler, users.js

Update users.js in **C:\>Nodejs\_WorkSpace\firstApplication\routes**.

*File: users.js*

```

router.get('/:id', function(req, res, next) {
 var user = users["user" + req.params.id]
 if(user) {
 res.send({ title: 'User Profile', user:user});
 }else{
 res.send({ Message: 'User not present'});
 }
}) ;

```

# Complete User.js

File: users.js

```
var express = require('express');
var router = express.Router();

var users = require('../users.json');

/* GET users listing. */
router.get('/', function(req, res) {
 res.send({ title: 'Users', users:users });
});

router.get('/:id', function(req, res, next) {
 console.log(req.params)
 var user = users["user" + req.params.id]
 if(user) {
 res.send({ title: 'User Profile', user:user});
 }else{
 res.send({ Message: 'User not present'});
 }
});

router.post('/addUser', function(req, res, next) {
 var body = '';
 req.on('data', function (data) {
 body += data;
 });
 req.on('end', function () {
 var json = JSON.parse(body);
 users["user"+json.id] = body;
 res.send({ Message: 'User Added'});
 });
});

module.exports = router;
```

## Output

We are using [Postman](#), a Chrome extension, to test our webservices.

Now run the app.js to see the result:

## Verify the Output. Server has started

Express server listening on port 3000

Make a request to firstApplication to get list of all the users. Put <http://localhost:3000/users> in POSTMAN with GET request and see the below result.

The screenshot shows the Postman interface. At the top, there are tabs for 'Normal', 'Basic Auth', 'Digest Auth', 'OAuth 1.0', and 'No environment'. Below that, the URL is set to 'http://localhost:3000/users', the method is 'GET', and there are buttons for 'Send', 'Preview', 'Add to collection', and 'Reset'. The 'Body' tab is selected, showing the response headers ('Headers (6)') and the status ('STATUS 200 OK TIME 296 ms'). Below the headers, there are buttons for 'Pretty', 'Raw', 'Preview', 'JSON', and 'XML'. The 'Pretty' button is highlighted. The response body is displayed as a JSON object with 20 numbered lines:

```

1 {
2 "title": "Users",
3 "users": [
4 "user1": {
5 "name": "mahesh",
6 "password": "password1",
7 "profession": "teacher"
8 },
9 "user2": {
10 "name": "suresh",
11 "password": "password2",
12 "profession": "librarian"
13 },
14 "user3": {
15 "name": "ramesh",
16 "password": "password3",
17 "profession": "clerk"
18 }
19 }
20 }
```

Make a request to firstApplication to add a new user. Put <http://localhost:3000/users/addUser> in POSTMAN with POST request and see the below result.

Make sure to add json body in and select POST as method.

```
{"name": "rakesh", "password": "password4", "profession": "teacher", "id": 4}
```

Normal Basic Auth Digest Auth OAuth 1.0 No environment

http://localhost:3000/users/addUser POST URL params Headers (0)

URL Parameter Key	Value
form-data	x-www-form-urlencoded raw JSON

```
1 {"name": "rakesh", "password": "password4", "profession": "teacher", "id": 4}
```

**Send Preview Add to collection Reset**

Body Headers (6) STATUS 200 OK TIME 315 ms

Pretty Raw Preview JSON XML

```
1 {
2 "Message": "User Added"
3 }
```

Make a request to firstApplication to get a user. Put http://localhost:3000/users/1 in POSTMAN with GET request and see the below result.

Normal Basic Auth Digest Auth OAuth 1.0 No environment

http://localhost:3000/users/1 GET URL params Headers (0)

**Send Preview Add to collection Reset**

Body Headers (6) STATUS 200 OK TIME 100 ms

Pretty Raw Preview JSON XML

```
1 {
2 "title": "User Profile",
3 "user": {
4 "name": "mahesh",
5 "password": "password1",
6 "profession": "teacher",
7 "id": 1
8 }
9 }
```

## N O D E . - S C A L I A N P G P L I C A T I O N

As node runs in a single thread mode, it uses an event-driven paradigm to handle concurrency. It also facilitates creation of child processes to leverage parallel processing on multi-core cpu based systems.

Child processes always have three streams child.stdin, child.stdout, and child.stderr which may be shared with the stdio streams of the parent process. they may be a separate stream objects which can be piped to and from.

There are three major ways to create child process.

- **exec** - child\_process.exec method runs a command in a shell/console and buffers the output.
- **spawn** - child\_process.spawn launches a new process with a given command
- **fork** - The child\_process.fork method is a special case of the spawn to create Node processes.

## exec method

child\_process.exec method runs a command in a shell and buffers the output. It has the following signature

```
child_process.exec(command[, options], callback)
```

- **command** String The command to run, with space-separated arguments
- **options** Object
  - **cwd** String Current working directory of the child process
  - **env** Object Environment key-value pairs
  - **encoding** String *Default:'utf8'*
  - **shell** String Shell to execute the command  
with *Default:'/bin/sh'on UNIX,'cmd.exe'on Windows,The shell should understand the -c switch on UNIX or /s on Windows. On Windows, command line parsing should be compatible with cmd.exe.*
  - **timeout** Number *Default:0*
  - **maxBuffer** Number *Default:200\*1024*
  - **killSignal** String *Default:'SIGTERM'*
  - **uid** Number Sets the user identity of the process.
  - **gid** Number Sets the group identity of the process.
- **callback** Function called with the output when process terminates
  - **error** Error
  - **stdout** Buffer
  - **stderr** Buffer
- **Return:** ChildProcess object

exec returns a buffer with a max size and waits for the process to end and tries to return all the buffered data at once

## Example

Create two js file named worker.js and master.js in **C:\>Nodejs\_WorkSpace**.

*File: worker.js*

```
console.log("Child Process " + process.argv[2] + " executed.");
```

*File: master.js*

```
const fs = require('fs');
const child_process = require('child_process');

for(var i=0; i<3; i++) {
 var workerProcess = child_process.exec('node worker.js '+i,
 function (error, stdout, stderr) {
 if (error) {
```

```

 console.log(error.stack);
 console.log('Error code: '+error.code);
 console.log('Signal received: '+error.signal);
 }
 console.log('stdout: ' + stdout);
 console.log('stderr: ' + stderr);
});

workerProcess.on('exit', function (code) {
 console.log('Child process exited with exit code '+code);
});
}

```

Now run the master.js to see the result:

```
C:\Nodejs_WorkSpace>node master.js
```

Verify the Output. Server has started

```

Child process exited with exit code 0
stdout: Child Process 1 executed.

stderr:
Child process exited with exit code 0
stdout: Child Process 0 executed.

stderr:
Child process exited with exit code 0
stdout: Child Process 2 executed.

```

## spawn method

`child_process.spawn` method launches a new process with a given command. It has the following signature

```
child_process.spawn(command[, args][, options])
```

- **command** String The command to run
- **args** Array List of string arguments
- **options** Object
  - **cwd** String Current working directory of the child process
  - **env** Object Environment key-value pairs
  - **stdio** Array/String Child's stdio configuration
  - **customFds** Array Deprecated File descriptors for the child to use for stdio
  - **detached** Boolean The child will be a process group leader
  - **uid** Number Sets the user identity of the process.
  - **gid** Number Sets the group identity of the process.
- **Return:** ChildProcess object

spawn returns streams stdout & stderr and it should be used when the process returns large amount of data. spawn starts receiving the response as soon as the process starts executing.

## Example

Create two js file named worker.js and master.js in **C:\>Nodejs\_WorkSpace**.

*File: worker.js*

```
console.log("Child Process " + process.argv[2] + " executed.");
```

*File: master.js*

```
const fs = require('fs');
const child_process = require('child_process');

for(var i=0; i<3; i++) {
 var workerProcess = child_process.spawn('node', ['worker.js', i]);

 workerProcess.stdout.on('data', function (data) {
 console.log('stdout: ' + data);
 });

 workerProcess.stderr.on('data', function (data) {
 console.log('stderr: ' + data);
 });

 workerProcess.on('close', function (code) {
 console.log('child process exited with code ' + code);
 });
}
```

Now run the master.js to see the result:

```
C:\Nodejs_WorkSpace>node master.js
```

Verify the Output. Server has started

```
stdout: Child Process 0 executed.

child process exited with code 0
stdout: Child Process 2 executed.

child process exited with code 0
stdout: Child Process 1 executed.

child process exited with code 0
```

## fork method

child\_process.fork method is a special case of the spawn to create Node processes. It has the following signature

```
child_process.fork(modulePath[, args][, options])
```

- **modulePath** String The module to run in the child
- **args** Array List of string arguments
- **options** Object
  - **cwd** String Current working directory of the child process
  - **env** Object Environment key-value pairs
  - **execPath** String Executable used to create the child process
  - **execArgv** Array List of string arguments passed to the executable *Default:process.execArgv*
  - **silent** Boolean If true, stdin, stdout, and stderr of the child will be piped to the parent, otherwise they will be inherited from the parent, see the "pipe" and "inherit" options for spawn's stdio for more details *default is false*
  - **uid** Number Sets the user identity of the process.
  - **gid** Number Sets the group identity of the process.
- **Return:** ChildProcess object

fork returns object with a built-in communication channel in addition to having all the methods in a normal ChildProcess instance.

## Example

Create two js file named worker.js and master.js in **C:\>Nodejs\_WorkSpace**.

*File: worker.js*

```
console.log("Child Process "+ process.argv[2] +" executed.");
```

*File: master.js*

```
const fs = require('fs');
const child_process = require('child_process');

for(var i=0; i<3; i++) {
 var worker_process = child_process.fork("worker.js", [i]);

 worker_process.on('close', function (code) {
 console.log('child process exited with code ' + code);
 });
}
```

Now run the master.js to see the result:

C:\Nodejs\_WorkSpace>node master.js

## Verify the Output. Server has started

```
Child Process 0 executed.
Child Process 1 executed.
Child Process 2 executed.
child process exited with code 0
child process exited with code 0
child process exited with code 0
```

# HTML For Beginners

---

<http://www.wix.com/my-account/sites/>

Create your website easily without codes and programs.

## What is HTML?

HTML is a language for describing web pages.

- HTML stands for Hyper Text Markup Language
- HTML is a **markup** language
- A markup language is a set of markup **tags**
- The tags **describe** document content
- HTML documents contain HTML **tags** and plain **text**
- HTML documents are also called **web pages**

## HTML Page Structure

```
<html>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
</body>
</html>
```

## HTML Versions

Since the early days of the web, there have been many versions of HTML:

Version	Year
HTML	1991
HTML+	1993
HTML 2.0	1995

HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML5	2012

## FIRST HTML Code: Introduction

```
<html>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```



## My First Heading

My first paragraph.

## HTML Headings

HTML headings are defined with the `<h1>` to `<h6>` tags

```
<html>
<body>
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
</body>
</html>
```

**Result:**

**This is heading 1**

**This is heading 2**

**This is heading 3**

**This is heading 4**

**This is heading 5**

**This is heading 6**

**HTML Paragraphs**

HTML paragraphs are defined with the <p> tag.

```
<html>
<body>
<p>This is First paragraph.</p>
<p>This is Second paragraph.</p>
<p>This is Third paragraph.</p>
</body>
</html>
```

**Note:** Browsers automatically add an empty line before and after a paragraph.

**Result:**

This is First paragraph.

This is Second paragraph.

This is Third paragraph.

**HTML Links**

HTML links are defined with the <a> tag.

```
<html>
<body>

Facebook
```

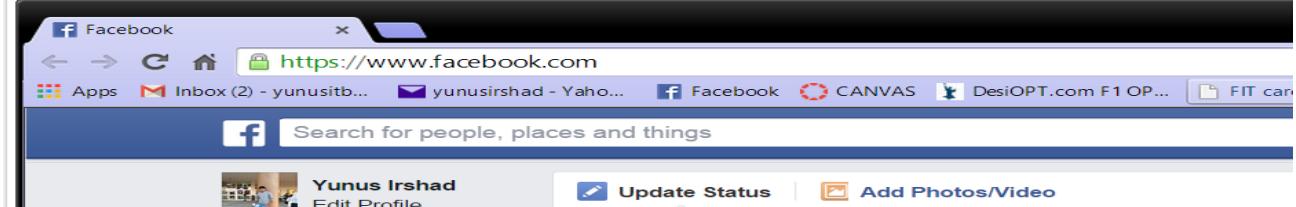
```
</body>
```

```
</html>
```

- If the link is in the same directory, declare (<a href = "XHTML Program.html"> This is XHTML program </a>)

### Result:

[Facebook](#)



## HTML Images

HTML images are defined with the <img> tag. You can find the width and height by right clicking the properties of the image..... RESIZE AN LARGE IMAGE use Photoshop.... But if you need to enter large image in small way in website ....It is best use in ebay and amazon.... **height = "30"**

```
<html>
```

```
<body>
```

```
</body>
```

```
</html>
```

### Result:



You can use the **style** attribute to specify the **width** and **height** of an image.

The values are specified in pixels (use px behind the value)

```

```

## Image Maps

For an image, you can create an image map, with clickable areas:

<p>Click on the sun or on one of the planets to watch it closer:</p>

```


<map name="planetmap">

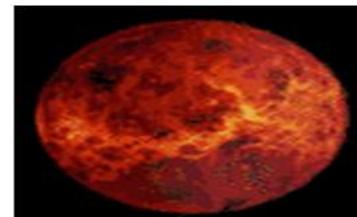
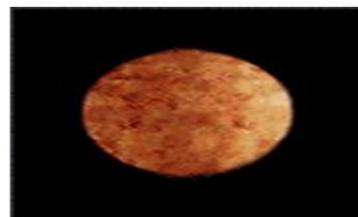
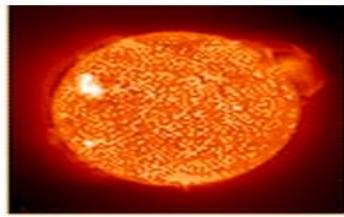
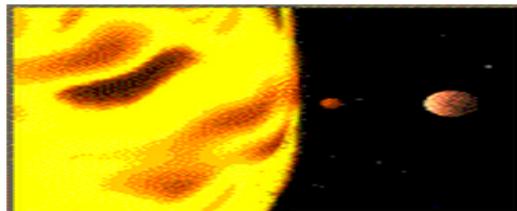
<area shape="rect" coords="0,0,82,126" alt="Sun" href="sun.htm">

<area shape="circle" coords="90,58,3" alt="Mercury" href="mercur.htm">

<area shape="circle" coords="124,58,8" alt="Venus" href="venus.htm">

</map>
```

**Click on the sun or on one of the planets to watch it closer:**



## Image Floating

You can let an image float to the left or right of a paragraph

```
<p>

A paragraph with an image. A paragraph with an image.

</p>
```



A paragraph with an image. A paragraph with an image.

## Empty HTML Elements ----- HTML Line Breaks

HTML elements with no content are called empty elements.

Use the <br> tag if you want a line break (a new line) without starting a new paragraph

```
<html>
<body>
<p>This is
a para
graph with line breaks</p>
</body>
</html>
```

## Result:

This is  
a para  
graph with line breaks

## Always Quote Attribute Values

Attribute values should always be enclosed in quotes.

Double style quotes are the most common, but single style quotes are also allowed.



**Tip:** In some rare situations, when the attribute value itself contains quotes, it is necessary to use single quotes: name='John "ShotGun" Nelson'

## HTML Lines

The <hr> tag creates a horizontal line in an HTML page.

```
<html>
<body>
<p>The hr tag defines a horizontal rule:</p>
<hr>
<p>This is a paragraph.</p>
<hr>
<p>This is a paragraph.</p>
</body>
</html>
```

**Result:**

The hr tag defines a horizontal rule:

---

This is a paragraph.

---

This is a paragraph.

**HTML <!--...--> Tag**

This is used for comments such as // in java.

```
<html>
<body>
<!-- This is a comment -->
<p>This is a paragraph.</p>
<!-- Comments are not displayed in the browser -->
</body>
</html>
```

**Invalid Comments**

<!-- This is not a valid comment -->

**HTML Output - Useful Tips**

You cannot be sure how HTML will be displayed. Large or small screens, and resized windows will create different results.

With HTML, you cannot change the output by adding extra spaces or extra lines in your HTML code.

The browser will remove extra spaces and extra lines when the page is displayed. Any number of lines count as one line, and any number of spaces count as one space.

```
<html>
<body>
<p>
 My Bonnie lies over the ocean.
 My Bonnie lies over the sea.
 My Bonnie lies over the ocean.
 Oh, bring back my Bonnie to me.
```

</p>

<p>Note that your browser ignores the layout in the HTML source code!</p>

</body>

</html>

## Result:

My Bonnie lies over the ocean. My Bonnie lies over the sea. My  
Bonnie lies over the ocean. Oh, bring back my Bonnie to me.

Note that your browser ignores the layout in the HTML source code!

## HTML Text Formatting Tags

Tag	Description
<b>	Defines bold text <b>James bond</b>
<em>	Defines emphasized text
<i>	Defines italic text <i>James bond</i>
<small>	Defines smaller text <small>James bond</small>
<strong>	Defines important text same as bold
<sub>	Defines subscripted text <u>This text contains</u> <sub>subscript</sub> text.
<sup>	Defines superscripted text <u>This text contains</u> <sup>superscript</sup> text.
<ins>	Defines inserted text <ins>red</ins>
<del> or <strike>	Defines deleted text <del>blue</del>
<mark>	Defines marked/highlighted text. <mark>Do not forget to buy</mark> <b>milk</b> today.
<tt>monospaced</tt>	The following word uses a monospaced typeface.
<big>big</big>	The following word uses a <big>big</big> typeface.

## <span> element

<p>This is the example of <span style="color:green">span tag</span>

This is the example of **span tag** and the **div tag** alongwith CSS

## HTML "Computer Output" Tags

Tag	Description	
<u>&lt;code&gt;</u>	Defines computer code text	Computer code
<u>&lt;kbd&gt;</u>	Defines keyboard text	Keyboard input
<u>&lt;samp&gt;</u>	Defines sample computer code	Sample text
<u>&lt;var&gt;</u>	Defines a variable	Computer variable
<u>&lt;pre&gt;</u> <pre> for i = 1 to 10 print i next i </pre>	Defines preformatted text  for i = 1 to 10 print i next i	

## HTML Citations, Quotations, and Definition Tags

Tag	Description
<u>&lt;abbr&gt;</u>	Defines an abbreviation or acronym  <abbr title="World Health Organization">WHO</abbr> When you keep the cursor above WHO it displays the title.
<u>&lt;address&gt;</u>	Defines contact information for the author/owner of a document  Written by W3Schools.com <a href="#">Email us</a> Address: Box 564, Disneyland Phone: +12 34 56 78
<u>&lt;bdo&gt;</u>  <bdo dir="rtl"> Here is some Hebrew text </bdo>	Defines the text direction Rtl is nothing but right to left direction.  displays as txet werbeH emos si ereH
<u>&lt;blockquote&gt;</u>	Defines a section that is quoted from another source
<u>&lt;q&gt;</u>	Defines an inline (short) quotation
<u>&lt;cite&gt;</u>	Defines the title of a work  <cite>The Scream</cite> by Edward Munch. Painted in 1893.



*The Scream* by Edward Munch. Painted in 1893.

<u>&lt;dfn&gt;</u>	Defines a definition term
<dfn>Definition term</dfn> 	<b>Definition term</b>

#### Code:

```
<body>
<h2>The blockquote Element</h2>
<p>The blockquote element specifies a section that is
quoted from another source.</p>
<p>Here is a quote from WWF's website:</p>
<blockquote
cite="http://www.worldwildlife.org/who/index.html">
For 50 years, WWF has been protecting the future of
nature. The world's leading conservation organization, WWF
works in 100 countries and is supported by 1.2 million
members in the United States and close to 5 million
globally.
</blockquote>
<p>Note: Browsers usually indent blockquote
elements.</p>
```

#### Result:

### The **blockquote** Element

The blockquote element specifies a section that is quoted from another source.

Here is a quote from WWF's website:

For 50 years, WWF has been protecting the future of nature. The world's leading conservation organization, WWF works in 100 countries and is supported by 1.2 million members in the United States and close to 5 million globally.

**Note:** Browsers usually indent blockquote elements.

```
<h2>The q Element</h2>
<p>The q element defines a short quotation.</p>

<p>WWF's goal is to:
<q>Build a future where people live in harmony with
nature.</q>
We hope they succeed.</p>
<p>Note: Browsers insert quotation marks around the
q element.</p>
```

### The **q** Element

The q element defines a short quotation.

WWF's goal is to: "Build a future where people live in harmony with nature." We hope they succeed.

**Note:** Browsers insert quotation marks around the q element.

## HTML Meta Tags

You can use <meta> tag to specify important keywords related to the document and later these keywords are used by the search engines while indexing your webpage for searching purpose.

Following is an example where we are adding HTML, Meta Tags, Metadata as important keywords about the document.

```
<meta name="keywords" content="HTML, Meta Tags, Metadata" />
<meta name="description" content="Learning about Meta Tags." />
```

## HTML Links

### How to use an image as a link.

<p>Create a link of an image:

```
</p>
```

<p><b>Note:</b> For IE 9 and earlier versions, the image-link above will show a border around the image. To remove the border around the image, add style="border:0;" to the img element.</p>

<p>Image-link: Still a link, but with no borders:

```
</p>
```

#### Result:

Create a link of an image:



**Note:** For IE 9 and earlier versions, the image-link above will show a border around the image. To remove the border around the image, add style="border:0;" to the img element.

Image-link: Still a link, but with no borders:



### HTML Links - The id Attribute

The id attribute can be used to create a bookmark inside an HTML document.

**Tip:** Bookmarks are not displayed in any special way. They are invisible to the reader.

#### Example

An anchor with an id inside an HTML document:

```
Useful Tips Section or
```

#### The title Attribute

HTML paragraphs are defined with the <p> tag.

In this example, the <p> element has a **title** attribute. The value of the attribute is "About W3Schools":

```
<p title="About W3Schools">
```

When you move the mouse over the element, the title will be displayed as a tooltip.

Create a link to the "Useful Tips Section" inside the same document:

```
Visit the Useful Tips Section
```

Or, create a link to the "Useful Tips Section" from another page:

```

Visit the Useful Tips Section
```

## **Link to a location on the same page**

How to link to a bookmark.

<b>Code:</b> <pre>&lt;p&gt; &lt;a href="#C3"&gt;See also Chapter 3.&lt;/a&gt; &lt;/p&gt;  &lt;h2&gt;&lt;a id="C1"&gt;Chapter 1&lt;/h2&gt; &lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;  &lt;h2&gt;&lt;a id="C2"&gt;Chapter 2&lt;/h2&gt; &lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;  &lt;h2&gt;&lt;a id="C3"&gt;Chapter 3&lt;/h2&gt; &lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;  &lt;h2&gt;&lt;a id="C4"&gt;Chapter 4&lt;/a&gt;&lt;/h2&gt; &lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;</pre>	<b>Result:</b> <p><a href="#">See also Chapter 3.</a></p> <p><b>Chapter 1</b> This chapter explains ba bla bla</p> <p><b>Chapter 2</b> This chapter explains ba bla bla</p> <p><b>Chapter 3</b></p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## **Break out of a frame**

How to break out of a frame (if your site is locked in a frame).

```
<p>Locked in a frame?</p>
```

```
Click here!
```

Closes this frame and opens the w3schools.com frame ..... in java we text dispose();

_blank	Opens the linked document in a new window or tab
_self	Opens the linked document in the same frame as it was clicked (this is default)
_parent	Opens the linked document in the parent frame
_top	Opens the linked document in the full body of the window
framename	Opens the linked document in a named frame

## **Result:**

Locked in a frame?

[Click here!](#)

## Create a mailto link

How to link to a mail message (will only work if you have mail installed).

<p>

This is an email link:

```

```

Send Mail</a>

</p>

### Result:

This is an email link: [Send Mail](#)

**Note:** Spaces between words should be replaced by %20 to ensure that the browser will display the text properly.

## Create a mailto link 2

Another mailto link.

<p>

This is another mailto link:

```
Send mail!
```

</p>

### Result:

This is another mailto link: [Send mail!](#)

**Note:** Spaces between words should be replaced by %20 to ensure that the browser will display the text properly.

## HTML Styles - CSS

<html>

<head>

<style>

```
body {background-color: lightgreen}
```

```
h1{color: red}
```

```

p {color: blue}

p{line-height:130%} // line spacing

</style>

</head>

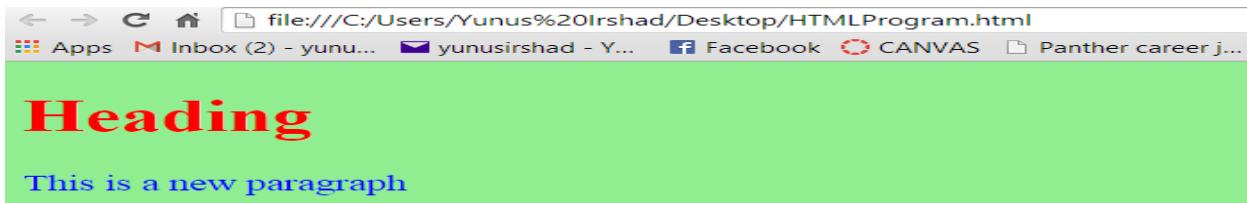
<h1>Heading</h1>

<p>This is a new paragraph</p>

</body>

</html>

```



## CSS styling can be added to HTML elements the following 3 ways:

- Inline - using the style **attribute** in HTML elements
- Internal - using the **<style>** element in the **<head>** section
- External - using external CSS **files**

### Internal Style Sheet

If you want to apply Style Sheet rules to a single document only then you can include those rules in header section of the HTML document using **<style>** tag. Rules defined in internal style sheet overrides the rules defined in an external CSS file.

Let's re-write above example once again, but here we will write style sheet rules in the same HTML document using **<style>** tag:

```

<style type="text/css">

 red{
 color: red;
 }

 .thick{
 font-size:20px;
 }

 .green{
 color:green;
 }

```

```

</style>
</head>
<body>

This is red

This is thick

This is green

This is thick and green

</body>
</html>

```

This will produce following result:

This is red

This is thick

This is green

This is thick and green

## Inline Styles

This line styling changes the text color and the left margin of single paragraph:

```
<p style="color:black; margin-left:20px;">This is a paragraph</p>
```



This is a paragraph

## CSS Font Family

```
<p style="font-family: courier">This is a Font paragraph</p>
```

This is a Font paragraph

## CSS Font Size

```
<p style="font-size: 300%">This is new font size</p>
```

## Font Size

```
Font size="1"

```

# This is new font size

## The <basefont> Element:

The <basefont> element is supposed to set a default font size, color, and typeface for any parts of the document that are not otherwise contained within a <font> tag. You can use the <font> elements to override the <basefont> settings.

The <basefont> tag also takes color, size and face attributes and it will support relative font setting by giving size a value of +1 for a size larger or -2 for two sizes smaller.

```
<basefont face="arial, verdana, sans-serif" size="2" color="#ff0000">
<p>This is the page's default font.</p>
<h2>Example of the <basefont> Element</h2>
<p>
This is darkgray text with two sizes larger
</p>
```

This is the page's default font.

**Example of the <basefont> Element**

This is darkgray text with two sizes larger

It is a courier font, a size smaller and black in color.

## CSS Text Alignment

```
<p style="text-align: left">This is left alignment</p>
<p style="text-align: right">This is right alignment</p>
<p style="text-align: center">This is center alignment</p>
```

This is left alignment

This is center alignment

This is right alignment

## External CSS

External style sheet are ideal when the style is applied to many pages.

With external style sheets, you can change the look of an entire site by changing one file.

**External styles** are defined in the **<head>** section of an HTML page, in the **<link>** element

```
<head>
<link rel="stylesheet" href="styles.css">
</head>
```

## I am formatted with an external style sheet

Me too!

## HTML Links

Remove the underline from this link:

```
<p>HTML Images</p>
```

Remove the underline from this link:

HTML Images

Tip: Use an inline style.

Remove the underline from this link:

HTML Images

Tip: Use an inline style.

## HTML Tables

```
<table border="1" style= "width:50%">
<th> CONTACTS </th>
<tr> <td>yunus</td>
 <td>irshad</td>
 <td>95</td>
</tr>
<tr> <td>zakira</td>
 <td>banu</td>
 <td>85</td>
</tr>
</table>
```

CONTACTS		
yunus		irshad
zakira	banu	95
		85

Tables are defined with the `<table>` tag.

Tables are divided into **table rows** with the `<tr>` tag.

Table rows are divided into **table data** with the `<td>` tag.

A table row can also be divided into **table headings** with the `<th>` tag.

A border can be added using the `border` attribute

To add borders, use the **CSS border** property:

```
table, th, td {
 border: 1px solid black;
}
```

## HTML Table with Collapsed Borders

If you want the borders to collapse into one border, add **CSS border-collapse**:

```
table, th, td {
 border: 1px solid black;
 border-collapse: collapse;
}
```

**Result:**

Jill	Smith	50
Eve	Jackson	94
John	Doe	80

## HTML Table with Cell Padding

Cell padding specifies the space between the cell content and its borders.

```
<table cellpadding = "10">>
```

Cell spacing specifies the space between the cell and its borders.

```
<table cellspacing = "10">>
```

If you do not specify a padding, the table cells will be displayed without padding.

To set the padding, use the **CSS padding** property:

```
table, th, td {
 border: 1px solid black;
 border-collapse: collapse;
}
th,td {
 padding: 15px;
}
```

### Result:

Jill	Smith	50
Eve	Jackson	94
John	Doe	80

## HTML Table with Border Spacing

Border spacing specifies the space between the cells.

To set the border spacing for a table, use the **CSS border-spacing** property:

```
table {
 border-spacing: 5px;
}
```

### Result:

Jill	Smith	50
Eve	Jackson	94
John	Doe	80

## Table Cells that Span Many Columns

To make a cell span more than one column, use the **colspan** attribute:

```
<table style="width:100%">
<tr>
```

```

<th>Name</th>
<th colspan="2">Telephone</th>
</tr>
<tr>
<td>Bill Gates</td>
<td>555 77 854</td>
<td>555 77 855</td>
</tr>
</table>

```

## Cell that spans two columns:

Name	Telephone
Bill Gates	555 77 854      555 77 855

## Table Cells that Span Many Rows

To make a cell span more than one row, use the **rowspan** attribute:

```

<table style="width:100%">
<tr>
<th>First Name:</th>
<td>Bill Gates</td>
</tr>
<tr>
<th rowspan="2">Telephone:</th>
<td>555 77 854</td>
</tr>
<tr>
<td>555 77 855</td>
</tr>
</table>

```

## Cell that spans two rows:

First Name:	Bill Gates
Telephone:	555 77 854 555 77 855

## HTML Table With a Caption

To add a caption to a table, use the **<caption>** tag:

```
<table style="width:100%">
<caption>Monthly savings</caption>
```

Monthly savings	
Month	Savings
January	\$100
February	\$50

## Tables IMAGE Backgrounds

```
<table border="1" bordercolor="green" background="/images/test.png">
```

Column 1	Column 2	Column 3
Row 1 Cell 1	Row 1 Cell 2	Row 1 Cell 3
Row 2 Cell 1	Row 2 Cell 2	Row 2 Cell 3
Row 3 Cell 1		

## Different Styles for Different Tables

Most of the examples above use a style attribute (width="100%") to define the width of each table.

This makes it easy to define different widths for different tables.

The styles in the <head> section, however, define a style for all tables in a page.

To define different styles for different tables, add a class attribute to the table:

```
<table class="names">
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Points</th>
</tr>
<tr>
<td>Eve</td>
<td>Jackson</td>
<td>94</td>
</tr>
</table>
```

Now you can define a different styles for this table:

```
table.names {
 width: 100%;
 background-color: #f1f1c1;
}
```

And add more styles:

```
table.names tr:nth-child(even) {
 background-color: #f1f1c1;
}
table.names tr:nth-child(odd) {
 background-color: #ffffff;
}
table.names th {
 color: white;
 background-color: #333333;
}
```

## Result:

<b>First Name</b>	<b>Last Name</b>	<b>Points</b>
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

<b>First Name</b>	<b>Last Name</b>	<b>Points</b>
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

## HTML Lists

### Unordered HTML Lists

An unordered list starts with the `<ul>` tag. Each list item starts with the `<li>` tag.

The list items will be marked with bullets (small black circles).

```

Coffee
Tea
```

```
Milk

```

### Result:

## An Unordered List:

- Coffee
- Tea
- Milk

## Unordered HTML Lists - The Style Attribute

A **style** attribute can be added to an **unordered list**, to define the style of the marker:

```
<ul style="list-style-type:square">
Coffee
Tea
Milk

```

### Square bullets

- Apples
- Bananas
- Lemons
- Oranges

### Circle bullets

- Apples
- Bananas
- Lemons
- Oranges

Style	Description
list-style-type:disc	The list items will be marked with bullets (default)
list-style-type:circle	The list items will be marked with circles
list-style-type:square	The list items will be marked with squares

## Ordered HTML Lists

An ordered list starts with the **<ol>** tag. Each list item starts with the **<li>** tag.

The list items will be marked with numbers.

```

Coffee
Milk

```

## An Ordered List:

1. Coffee
2. Tea
3. Milk

```
<ol type="i" start="4">
```

Starts with 4.Coffee 5. Tea 6. Milk

## HTML Ordered Lists - The Type Attribute

A **type** attribute can be added to an **ordered list**, to define the type of the marker:

```
<ol type="A">
Coffee
Tea
Milk

```

Type	Description
type="1"	The list items will be numbered with numbers (default)
type="A"	The list items will be numbered with uppercase letters
type="a"	The list items will be numbered with lowercase letters
type="I"	The list items will be numbered with uppercase roman numbers
type="i"	The list items will be numbered with lowercase roman numbers

## HTML Description Lists

A description list, is a list of terms, with a description of each term.

The **<dl>** tag defines a description list.

The **<dt>** tag defines the term (name), and the **<dd>** tag defines the data (description).

```
<dl>
<dt>Coffee</dt>
<dd>- black hot drink</dd>
<dt>Milk</dt>
```

```
<dd>- white cold drink</dd>
</dl>
```

## A Description List:

```
Coffee
 - black hot drink
Milk
 - white cold drink
```

## Nested HTML Lists

List can be nested (lists inside lists).

```

Coffee
Tea

 Black tea
 Green tea

Milk

```

## A nested List:

- Coffee
- Tea
  - Black tea
  - Green tea
- Milk

## HTML Layouts

### HTML Layout Using <div> Elements

The div element is a block level element, designed for grouping HTML elements.

But layout can be designed using <div> elements, because CSS can position and style <div> elements.

The following example uses five <div> elements to create a multiple column layout:

```
<div id="container" style="width:500px">

<div id="header" style="background-color:#FFA500;">
<h1 style="margin-bottom:0;">Main Title of Web Page</h1></div>

<div id="menu" style="background-color:#FFD700;height:200px;width:100px;float:left;">
```

```
Menu

HTML

CSS

JavaScript</div>
```

```
<div id="content" style="background-color:#EEEEEE;height:200px;width:400px;float:left;">
Content goes here</div>
```

```
<div id="footer" style="background-color:#FFA500;clear:both;text-align:center;">
Copyright © W3Schools.com</div>
```

```
</div>
```

## HTML Layout Using Tables

The purpose of the `<table>` element is to display tabular data.

But layout can be designed using `<table>` element, because all table elements can be styled with CSS.

The following example uses a table with 3 rows and 2 columns - the first and last row spans 2 columns using the `colspan` attribute:

```
<table style="width:500px;" cellpadding="0" cellspacing="0">
<tr>
<td colspan="2" style="background-color:#FFA500;">
<h1 style="margin:0;padding:0;">Main Title of Web Page</h1>
</td>
</tr>

<tr>
<td style="background-color:#FFD700;width:100px;vertical-align:top;">
Menu

HTML

CSS

JavaScript
</td>
<td style="background-color:#eeeeee;height:200px;width:400px;vertical-align:top;">
Content goes here</td>
</tr>

<tr>
<td colspan="2" style="background-color:#FFA500;text-align:center;">
Copyright © W3Schools.com</td>
</tr>
```

</table>

**Result:**

# Main Title of Web Page

**Menu**

HTML

CSS

JavaScript

Content goes here

Copyright © W3Schools.com

## HTML Forms

HTML forms are used to pass data to a server.

An HTML form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements.

The <form> tag is used to create an HTML form:

```
<form>
 ...
</form>
```

## HTML Forms - Text Fields

<input type="text"> defines a one-line input field that a user can enter text into:

```
<form>
 First name: <input type="text" name="firstname">

 Last name: <input type="text" name="lastname">
</form>
```

How the HTML code above looks in a browser:

First name:

Last name:

**Note:** The form itself is not visible. Also note that the default width of a text field is 20 characters.

<b>Maxlength="5"</b>	Allows to specify the maximum number of characters a user can enter into the text box.
----------------------	----------------------------------------------------------------------------------------

## Password Field

<input type="password"> defines a password field:

```
<form>
 Password: <input type="password" name="pwd">
</form>
```

How the HTML code above looks in a browser:

Password:

**Note:** The characters in a password field are masked (shown as asterisks or circles).

## TEXT AREA:

```
<form>
 Description :

 <textarea rows="5" cols="50" name="description">
 Enter description here...
 </textarea>
</form>
```

Description :

## Radio Buttons

<input type="radio"> defines a radio button. Radio buttons let a user select ONLY ONE of a limited number of choices:

```
<form>
 <input type="radio" name="sex" value="male">Male

```

```
<input type="radio" name="sex" value="female">Female
</form>
```

How the HTML code above looks in a browser:

- Male
- Female

## Checkboxes

`<input type="checkbox">` defines a checkbox. Checkboxes let a user select ZERO or MORE options of a limited number of choices.

```
<form>
<input type="checkbox" name="vehicle" value="Bike">I have a bike

<input type="checkbox" name="vehicle" value="Car">I have a car
</form>
```

How the HTML code above looks in a browser:

- I have a bike
- I have a car

## Submit Button

`<input type="submit">` defines a submit button.

A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute. The file defined in the action attribute usually does something with the received input:

```
<form name="input" action="demo_form_action.asp" method="get">
Username: <input type="text" name="user">
<input type="submit" value="Submit">
</form>
```

How the HTML code above looks in a browser:

Username:

```
<input type="reset" name="reset" value="Reset" />
<input type="button" name="ok" value="OK" />
<input type="image" name="imagebutton" src="/html/images/logo.png" />
```

This will produce following result:



## HTML HIDDEN form controls:

```
<p>This is page 10</p>
<input type="hidden" name="pagename" value="10" />
```

This will produce following result:

This is page 10

## HTML <fieldset> and <legend> Tag

Group related elements in a form:

```
<form>
 <fieldset>
 <legend>Personalia:</legend>
 Name: <input type="text">

 Email: <input type="text">

 Date of birth: <input type="text">
 </fieldset>
</form>
```

### Result:

Personalia:

Name:	<input type="text"/>
Email:	<input type="text"/>
Date of birth:	<input type="text"/>

## HTML <select> or <option> Tag

Create a drop-down list with four options:

```
<select>
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="mercedes">Mercedes</option>
<option value="audi">Audi</option>
</select>
```

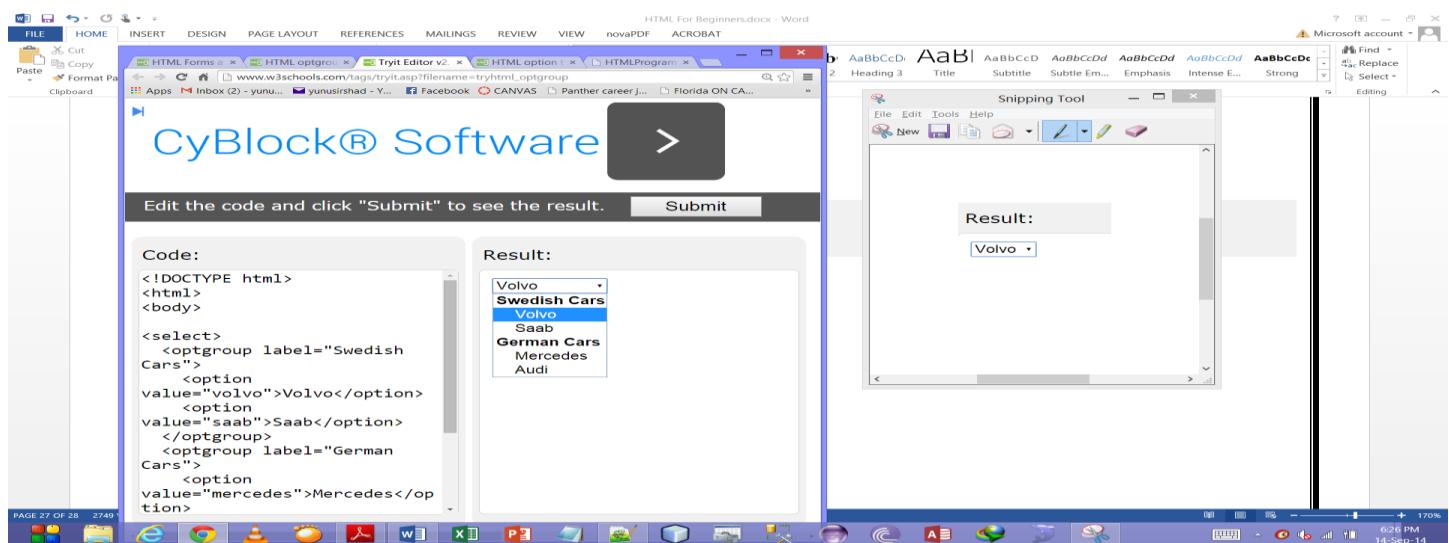
**Result:**

Volvo ▾

## HTML <optgroup> Tag

Group related options with <optgroup> tags:

```
<select>
<optgroup label="Swedish Cars">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
</optgroup>
<optgroup label="German Cars">
<option value="mercedes">Mercedes</option>
<option value="audi">Audi</option>
</optgroup>
</select>
```



## HTML FILE UPLOAD BOX BROWSE:

```
<form>
<input type="file" name="fileupload" accept="image/*" />
</form>
```

This will produce following result:

## HTML Iframes

An iframe is used to display a web page within a web page.

### Syntax for adding an iframe:

```
<iframe src="URL"></iframe>
```

The URL points to the location of the separate page.

## Iframe - Set Height and Width

The height and width attributes are used to specify the height and width of the iframe.

The attribute values are specified in pixels by default, but they can also be in percent (like "80%").

```
<iframe src="demo_iframe.htm" width="200" height="200"></iframe>
```

Result:

[W3Schools.com](#)

## Iframe - Remove the Border

The frameborder attribute specifies whether or not to display a border around the iframe.

Set the attribute value to "0" to remove the border:

```
<iframe src="demo_iframe.htm" frameborder="0"></iframe>
```

Result:

This page is displayed  
in an iframe

## Use iframe as a Target for a Link

An iframe can be used as the target frame for a link.

The target attribute of a link must refer to the name attribute of the iframe:

```
<iframe src="demo_iframe.htm" name="iframe_a"></iframe>
<p>W3Schools.com</p>
```

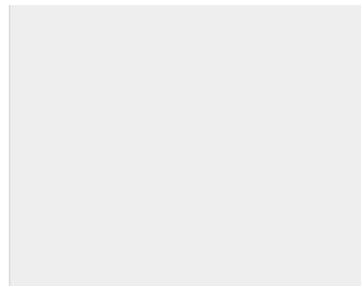
This page is displayed  
in an iframe

[W3Schools.com](http://www.w3schools.com)

## CREATE FRAMES:-

```
<frameset rows="10%,80%,10%">
 <frame name="top" src="/html/top_frame.htm" />
 <frame name="main" src="/html/main_frame.htm" />
 <frame name="bottom" src="/html/bottom_frame.htm" />
<noframes>
<body>
 Your browser does not support frames.
</body>
</noframes>
</frameset>
```

This will produce following result:



## FRAME SET SIZE in WEB PAGE

`<frameset cols="200, *">` //200 is the size of the column on left side and \* is the balance size of the page  
`<frame src="/html/menu.htm" name="menu_page" />`  
`<frame src="/html/main.htm" name="main_page" />`



## HTML URL:

A URL is another word for a web address.

A URL can be composed of words (w3schools.com), or an Internet Protocol (IP) address (192.68.20.50).

Most people enter the name when surfing, because names are easier to remember than numbers.

### URL - Uniform Resource Locator

Web browsers request pages from web servers by using a URL.

When you click on a link in an HTML page, an underlying `<a>` tag points to an address on the web.

A Uniform Resource Locator (URL) is used to address a document (or other data) on the web.

A web address, like `http://www.w3schools.com/html/default.asp` follows these syntax rules:

**scheme://host.domain:port/path/filename**

Explanation:

- **scheme** - defines the type of Internet service (most common is **http**)
- **host** - defines the **domain host** (default host for http is **www**)

- **domain** - defines the Internet **domain name** (w3schools.com)
- **port** - defines the **port number** at the host (default for http is **80**)
- **path** - defines a **path** at the server (If omitted: the root directory of the site)
- **filename** - defines the name of a document or resource

## Common URL Schemes

The table below lists some common schemes:

Scheme	Short for	Used for
http	HyperText Transfer Protocol	Common web pages. Not encrypted
https	Secure HyperText Transfer Protocol	Secure web pages. Encrypted
ftp	File Transfer Protocol	Downloading or uploading files
file		A file on your computer

## ENABLE AND DISABLE INPUTS:

```
<input checked="checked">
<input readonly="readonly">
<input disabled="disabled">
```



## HTML Embed Multimedia

Sometimes you need to add music or video into your web page. The easiest way to add video or sound to your web site is to include the special HTML tag called **<embed>**. This tag causes the browser itself to include controls for the multimedia automatically provided browser supports **<embed>** tag and given media type.

You can also include a **<noembed>** tag for the browsers which don't recognize the **<embed>** tag. You could, for example, use **<embed>** to display a movie of your choice, and **<noembed>** to display a single JPG image if browser does not support **<embed>** tag.

```
<embed src="video.mp4" width="100%" height="60" >
<noembed></noembed>
</embed>
```

Attribute	Description
align	Determines how to align the object. It can be set to either <i>center</i> , <i>left</i> or <i>right</i> .
autoplay	This boolean attribute indicates if the media should start automatically. You can set it either true or false.
loop	Specifies if the sound should be played continuously (set loop to true), a certain number of times (a positive value) or not at all (false)
playcount	Specifies the number of times to play the sound. This is alternate option for <i>loop</i> if you are using IE.
hidden	Specifies if the multimedia object should be shown on the page. A false value means no and true values means yes.
volume	Controls volume of the sound. Can be from 0 (off) to 100 (full volume).

## Background Audio

You can use HTML **<bgsound>** tag to play a soundtrack in the background of your webpage. This tag is supported by Internet Explorer only and most of the other browsers ignore this tag.

```
<bgsound src="theme.mp3">
</bgsound>
```

## HTML Object tag

HTML 4 introduces the **<object>** element, which offers an all-purpose solution to generic object inclusion. The **<object>** element allows HTML authors to specify everything required by an object for its presentation by a user agent

You can **embed an HTML** document in an HTML document itself as follows:

```
<object data="data/test.htm" type="text/html" width="300" height="200">
</object>
```

Here *alt* attribute will come into picture if browser does not support *object* tag.

You can **embed a PDF document** in an HTML document as follows:

```
<object data="data/test.pdf" type="application/pdf" width="300" height="200">
</object>
```

You can specify some parameters related to the document with the **<param> tag**. Here is an example to embed a wav file:

```
<object data="data/test.wav" type="audio/x-wav" width="200" height="20">
<param name="src" value="data/test.wav">
<param name="autoplay" value="false">
<param name="autoStart" value="0">
</object>
```

You can add **a flash document** as follows: just add embed

## HTML Marquees

An HTML marquee is a scrolling piece of text displayed either horizontally across or vertically down your webpage depending on the settings. This is created by using HTML <marquees> tag.

**Note:** The HTML <marquee> tag may not be supported by various browsers so it's not recommended to rely on this tag, instead you can use Javascript and CSS to create such effects.

```
<marquee attribute_name="attribute_value"more attributes>
```

One or more lines or text message or image

```
</marquee>
```

Attribute	Description
width	This specifies the width of the marquee. This can be a value like 10 or 20% etc.
height	This specifies the height of the marquee. This can be a value like 10 or 20% etc.

direction	This specifies the direction in which marquee should scroll. This can be a value like <i>up</i> , <i>down</i> , <i>left</i> or <i>right</i> .
behavior	This specifies the type of scrolling of the marquee. This can have a value like <i>scroll</i> , <i>slide</i> and <i>alternate</i> .
scrolldelay	This specifies how long to delay between each jump. This will have a value like 10 etc.
scrollamount	This specifies the speed of marquee text. This can have a value like 10 etc.
loop	This specifies how many times to loop. The default value is INFINITE, which means that the marquee loops endlessly.
bgcolor	This specifies background color in terms of color name or color hex value.
hspace	This specifies horizontal space around the marquee. This can be a value like 10 or 20% etc.
vspace	This specifies vertical space around the marquee. This can be a value like 10 or 20% etc.

## HTML Header

We have learnt that a typical HTML document will have following structure:

```
<html>
 <head>
 Document header related tags
 </head>
 <body>
 Document body related tags
 </body></html>
```

## The HTML <meta> Tag

The HTML <meta> tag is used to provide metadata about the HTML document which includes information about page expiry, page author, list of keywords, page description etc.

```
<!-- Provide list of keywords -->
<meta name="keywords" content="C, C++, Java, PHP, Perl, Python">
<!-- Provide description of the page -->
<meta name="description" content="Simply Easy Learning by Tutorials Point">
```

## The HTML <base> Tag

The HTML <base> tag is used for specifying the base URL for all relative URLs in a page, which means all the other URLs will be concatenated into base URL while locating for the given item.

For example, all the given pages and images will be searched after prefixing the given URLs with base URL <http://www.tutorialspoint.com/> directory:

```
<base href="http://www.tutorialspoint.com/" />
```

This will produce following result:



### HTML Tutorial

But if you change base URL to something else, for example, if base URL is <http://www.tutorialspoint.com/home> then image and other given links will become like <http://www.tutorialspoint.com/home/images/logo.png> and <http://www.tutorialspoint.com/home/html/index.htm>

## The HTML <link> Tag

The HTML <link> tag is used to specify relationships between the current document and external resource. Following is an example to link an external style sheet file available in **css** sub-directory within web root:

```
<link rel="stylesheet" type="text/css" href="/css/style.css">
```

## The HTML <style> Tag

The HTML <style> tag is used to specify style sheet for the current HTML document. Following is an example to define few style sheet rules inside <style> tag:

```

<style type="text/css">
.myclass{
 background-color: #aaa;
 padding: 10px;
}
</style>
<p class="myclass">Hello, World!</p>

```

## HTML Javascript

A **script** is a small piece of program that can add interactivity to your website. For example, a script could generate a pop-up alert box message, or provide a dropdown menu. This script could be written using Javascript or VBScript.

You can write various small functions, called event handlers using any of the scripting language and then you can trigger those functions using HTML attributes.

Now a days only **Javascript** and associated frameworks are being used by most of the web developers, VBScript is not even supported by various major browsers.

You can keep Javascript code in a separate file and then include it wherever it's needed, or you can define functionality inside HTML document itself. Let's see both the cases one by one with suitable examples.

## External Javascript

If you are going to define a functionality which will be used in various HTML documents then it's better to keep that functionality in a separate Javascript file and then include that file in your HTML documents. A Javascript file will have extension as **.js** and it will be included in HTML files using **<script>** tag.

Consider we define a small function using Javascript in **script.js** which has following code:

```

function Hello()
{
 alert("Hello, World");
}

```

Now let's make use of the above external Javascript file in our following HTML document:

```

<script src="/html/script.js" type="text/javascript"/></script>

```

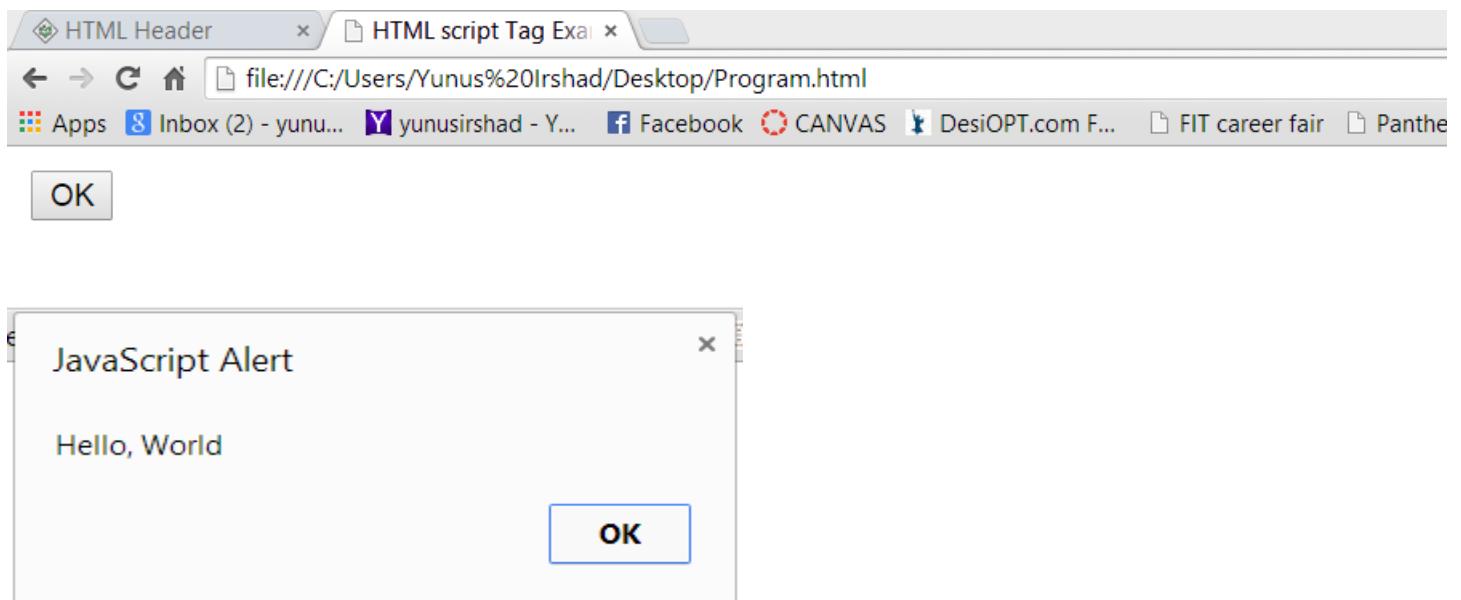
```
</head>
<body>
<input type="button" onclick="Hello();" name="ok" value="Click Me" />
```

## Internal Script

You can write your script code directly into your HTML document. Usually we keep script code in header of the document using `<script>` tag, otherwise there is no restriction and you can put your source code anywhere in the document but inside `<script>` tag.

```
<script type="text/javascript">
function Hello(){
 alert("Hello, World");
}
</script>
<body>
<input type="button" onclick="Hello();" name="ok" value="OK" />
</body>
```

This will produce following result, where you can try to click on the given button:



## Event Handlers

Event handlers are nothing but simply defined functions which can be called against any mouse or keyboard event. You can define your business logic inside your event handler which can vary from a single to 1000s of line code.

Following example explains how to write an event handler. Let's write one simple function `EventHandler()` in the header of the document. We will call this function when any user brings mouse over a paragraph.

```
<script type="text/javascript">
function EventHandler(){
 alert("I'm event handler!!");
}
</script>
</head>
<body>
<p onmouseover="EventHandler();">Bring your mouse here to see an alert</p>
</body>
</html>
```

## Hide Scripts from Older Browsers

Although most (if not all) browsers these days support Javascript, but still some older browsers don't. If a browser doesn't support JavaScript, instead of running your script, it would display the code to the user. To prevent this, you can simply place HTML comments around the script as shown below.

### JavaScript Example:

```
<script type="text/javascript">
<!--
document.write("Hello Javascript!");
//-->
</script>
```

### VBScript Example:

```
<script type="text/vbscript">
<!--
document.write("Hello VBScript!")
-->
```

```
'-->
```

```
</script>
```

## The <noscript> Element

You can also provide alternative info to the users whose browsers don't support scripts and for those users who have disabled script option their browsers. You can do this using the <noscript> tag.

```
<noscript>Your browser does not support Javascript!</noscript>
```

## Default Scripting Language

There may be a situation when you will include multiple script files and ultimately using multiple <script> tags. You can specify a default scripting language for all your *script* tags. This saves you from specifying the language everytime you use a script tag within the page. Below is the example:

```
<meta http-equiv="Content-Script-Type" content="text/JavaScript" />
```

Note that you can still override the default by specifying a language within the script tag.



# JavaScript Programming

## JAVASCRIPT:

JavaScript is a scripting language used to make webpages interactive.

```
<html><head>
 <title>JavaScript Program</title>
<head><body>
<script>
 document.write("Hello World"); // prints the hello world on the screen
</script>
</body></html>
```

- Comments are declared as same as JAVA “//” and for multiple comments “/\* \*/”



## VARIABLES:

```
<script>
 var apple = 26; // declaring an integer variable
 document.write(apple);

 var apple = " This is a String"; // declaring a character variable
 document.write(apple);

 var apple = " Yunus said, \"I love you\"";
 // use escape character if you have to specify string in quotations
 document.write(apple);

 var banana = apple; // if you don't use quotations in string it will take as a variable
 document.write(banana);
</script>
```



26 This is a String Yunus said, "I love you" Yunus said, "I love you"

## USING VARIABLES WITH STRING:

```
<script>

 var name = "yunus";
 var age = "23";
 document.write("My name is "+name+" and my age is "+age); // same as java

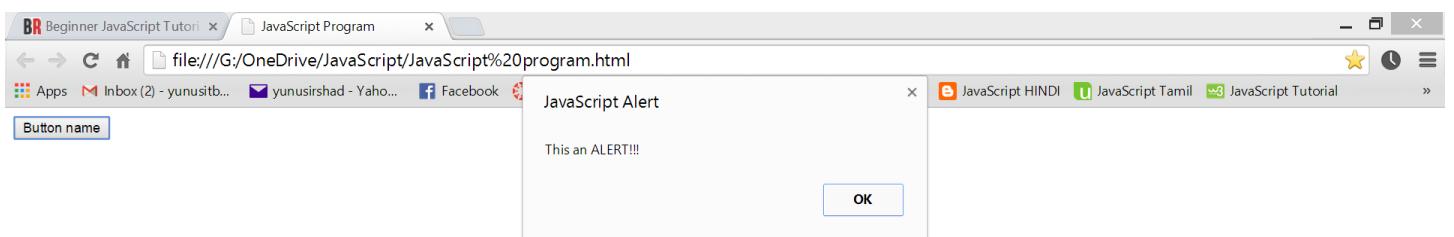
</script>
```

## FUNCTIONS:

```
<script>

 function funky() // function name is created
 {
 alert("This an ALERT!!!"); // an alert message is shown
 }
 // you can use this funky(); which calls the function

</script>
<form>
 <input type="button" value="Button name" onclick="funky()"> // on click calls the function
</form>
```



## USING PARAMETERS WITH FUNCTIONS:

```
<script>

 function funky(name) // parameter is assigned
 {
 alert("My name is "+name);
 }
```

```

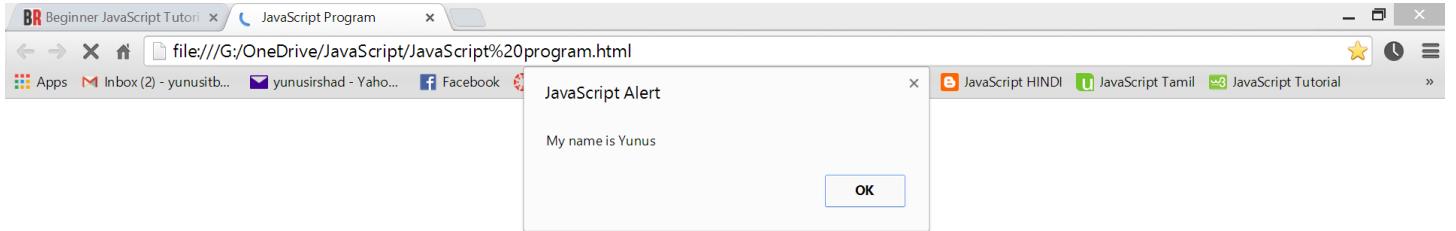
}

funky("Yunus"); // function has a parameter

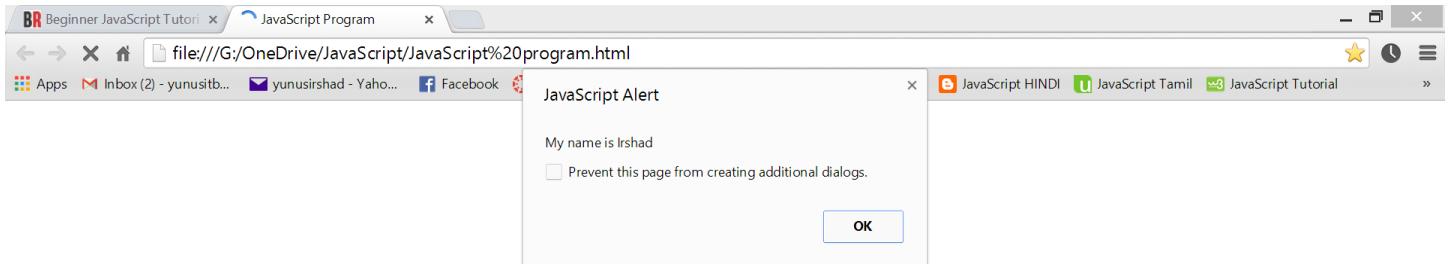
funky("Irshad"); // another function created if you want to display two windows or alerts

</script>

```



If you click OK...



## FUNCTIONS WITH MULTIPLE PARAMETERS:

```

<script>

 function funky(name,age) // multi parameter assigned
 {
 document.write("My name is "+name+" and my age is "+age+"
"); // breaks the line
 }

 funky("Yunus",24); // multi parameter function is called
 funky("Irshad",26);

</script>

```



## RETURN:

```

<script>

 function funky()
 {
 return "Dhoommeeeeeee"; //returning the value
 }

```

```

 }

 document.write(funky()); // calling the function

</script>

```

Dhoommeeee

```

<script>

 function funky(a,b)
 {
 var c = a+b;
 return c; // if you don't add return it displays "UNDEFINED"
 }

 document.write(funky(10,20));

</script>

```

30

## CALLING A FUNCTION FROM ANOTHER FUNCTION:

```

<script>

 function first() // first function declared
 {
 document.write("This is First Function");
 }

 function second() // second function declared
 {
 document.write("This is Second Function");
 }

 function start() // calling other functions
 {
 first();
 second();
 }

```

```

 }
 start(); // don't forget to call the function
</script>

```

This is First FunctionThis is Second Function

## GLOBAL/LOCAL VARIABLES:

```

<script>
 var name= "Yunus"; //this is global variable which is declared outside function.

 function funky()
 {
 document.write(name);
 }

 funky();
</script>

```

Yunus

```

<script>
 function funky()
 {
 var name2 = "Irshad"; //this is local variable which is declared inside function.

 document.write(name);
 document.write(name2);

 }

 funky();

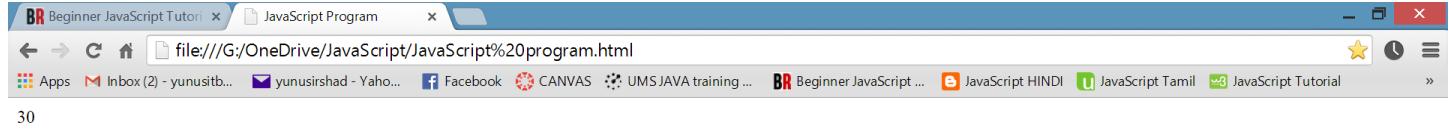
 document.write(name2); // name2 variable is not declared globally so it will not print
</script>

```

YunusIrshad

## ARITHMETIC—ASSIGNMENT OPERATORS:

```
<script>
 var apples = 10+20; // + is an addition operator
 document.write(apples);
// we have more math operators such as - (subtract), *(multiply), / (divide), %(remainder or modulus)
</script>
```



```
<script>
 var apples = 25;
 apples += 50;
// += is the assignment operator and it can be assigned to addition, multiply, subtract, divide.
 document.write(apples);
</script>
```



## IF & IF-ELSE STATEMENT:

```
<script>
 var apples = 25;
 if(apples == 25)
// checks the statement you can use <=(less than equal to), >=(greater than equal to), !=(not equal)
 {
 document.write("Yes the condition is true");
 // prints only if the condition satisfy the statement
 } else
 {
 document.write("No the condition is false");
 // prints only if the condition doesn't satisfy the statement
 }
```

```
</script>
```



Yes the condition is true

## NESTED STATEMENT:

```
<script>

 var firstname = "yunus";
 var lastname = "irshad";
 if(firstname=="yunus")
 {
 if(lastname=="irshad") // nested if statement
 {
 document.write("Yes he won the price");
 }
 }
</script>
```

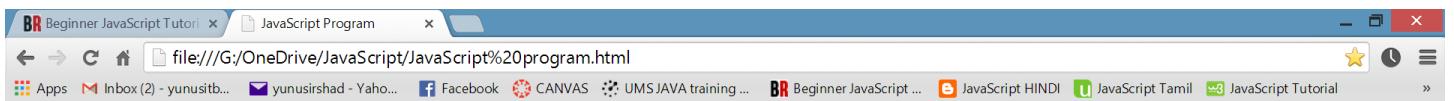


Yes he won the price

## AND OR OPERATORS USED IN STATEMENTS:

```
<script>

 var firstname = "yunus";
 var lastname = "irshad";
 // by this AND-OR operator eliminates the nested statements
 if((firstname=="yunus") && (lastname=="irshad"))
 // AND "&&"operator in which both conditions must be true
 {
 // OR "||"operator in which any one condition must be true
 document.write("Yes he won the prices");
 }
</script>
```



Yes he won the prices

## SWITCH:

```
<script>

 var name="yunus";

 switch(name) // checks the statement

 {

 case "yunus": document.write("Yes he is "+name);

 break;

 case "irshad": document.write("Yes he is "+name);

 break;

 default: document.write("INVALID NAME"); // this is a default statement if doesn't matches

 }

</script>
```



Yes he is yunus

## FOR LOOP:

```
<script>

 for(x=0; x<10; x++) // in for loop you don't need to initialize the variable

 {

 document.write(x+"
"); // break line must in quotes

 }

</script>
```

## WHILE LOOP:

```
<script>

 var x=0;

 while(x<10)

 {

 document.write(x+"
");

 }
```

```

 x++;
 }

</script>

```

## DO-WHILE LOOP:

```

<script>

 var x=0;

 do
 {
 document.write(x+"
"); // break line must in quotes

 x++;
 }while(x<10);

</script>

```

```

0
1
2
3
4
5
6
7
8
9

```

## EVENT HANDLER:

- Event handlers are built in keywords...

```

<form>

 <input type="button" value="Button Name" onclick="alert('Hello JavaScript');alert('Hello again')">
 // Runs the javascript code here

 //alert() is a javascript and must end with semicolon, you can add as many javascripts after
 semicolon, contents in alert() displays in box

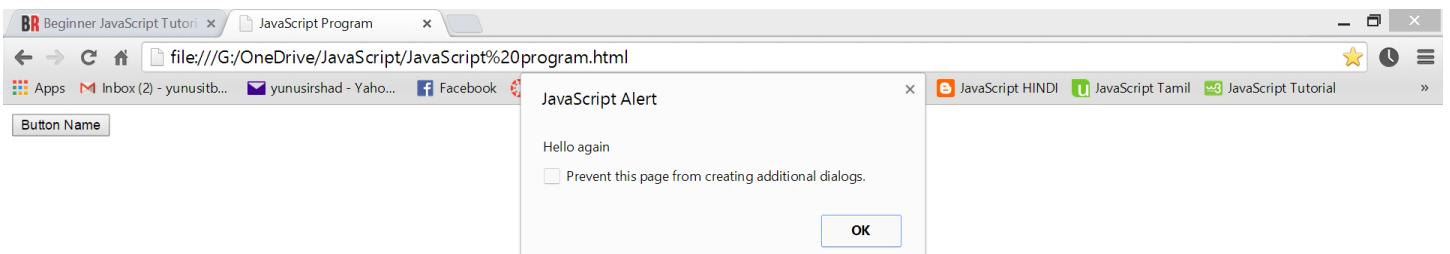
</form>

```

JavaScript Alert

Hello JavaScript

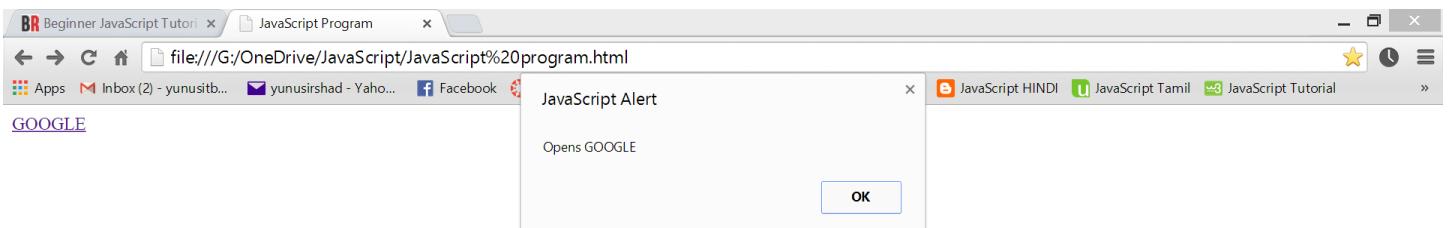
OK



## ONMOUSEOVER – ONMOUSEOUT:

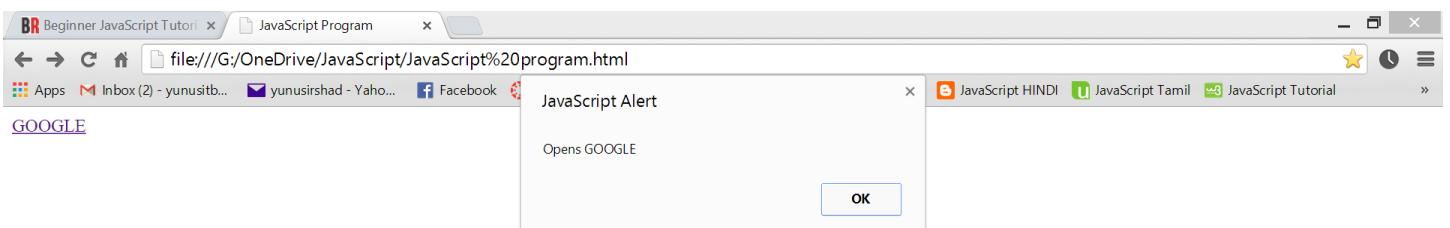
```
<body>
GOOGLE
</body>
```

**// if you take mouse over the hyperlink it displays the JavaScript**



```
<body>
GOOGLE
</body>
```

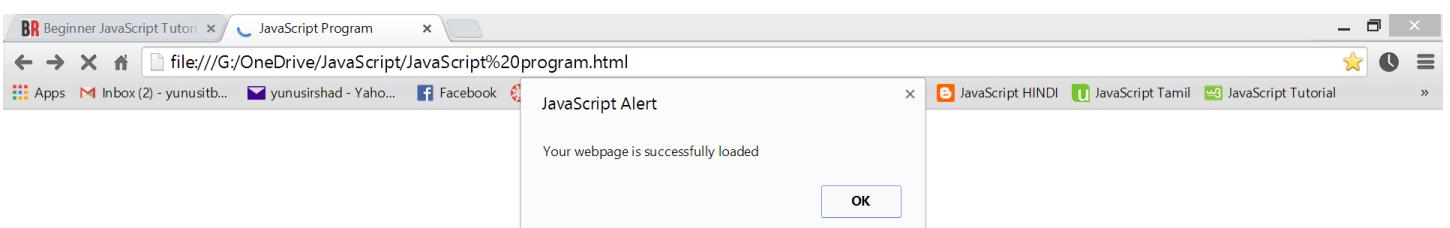
**// if you take mouse in and while coming out of the hyperlink it displays the JavaScript**



## ONLOAD – ONUNLOAD:

```
<body onload="alert('Your webpage is successfully loaded');">
</body>
```

**// displays a JavaScript when the page is loaded...**



```
<body onunload="alert('Goodbye');">
</body>
```

**// displays a good bye message when you exit the web page or browser**

## CREATING OBJECTS:

- Objects have two ----- properties associated to variables and methods.

```
<head><script>

 function person(name,age)

 {

 this.name = name; // this is a constructor for function

 this.age = age;

 }

 var object = new person("Yunus",23); // object is been created

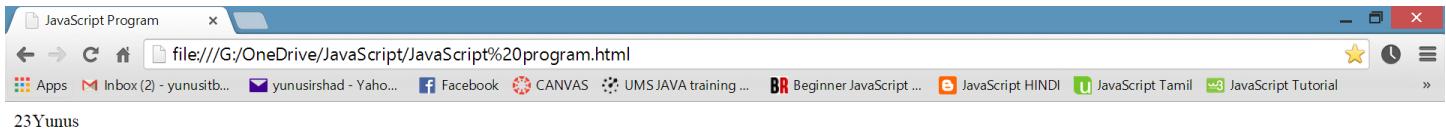
</script><head>

<body><script>

 document.write(object.age); // object with his property

 document.write(object.name);

</script></body>
```



## OBJECT INITIALIZERS:

```
<script>

 var object1 = {name:"Yunus", age:23}; // object initialization

 var object2 = {name:"Irshad", age:25};

</script><head>

<body><script>

 document.write("My name is "+object1.name+" and my another name is "+object2.name+" and my age is "+object1.age);

</script>
```



## ADDING METHODS TO OBJECTS:

```
<script>
```

```

function people(name,age)
{
 this.name = name;
 this.age = age;
 this.retire = yearsleft; // retire takes place with yearsleft method in constructor function
}

function yearsleft()
{
 var numberofyears = 50-this.age; //takes the age of the constructor
 return numberofyears;
}

var object1 = new people("Yunus",23);

</script><head>
<body><script>

 document.write(object1.retire()+" Years more for me to retire") //object takes the retire() method

</script>

```



## ARRAYS:

```

<script> //arrays are declared to list long datas like names, phone numbers.

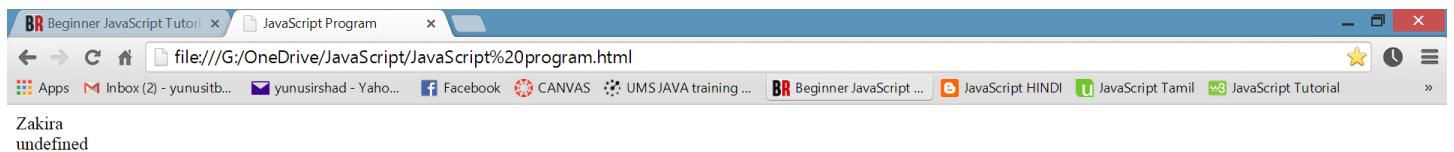
 var names = new Array("Yunus","Irshad","Zakira","Banu");

 document.write(names[2]+
); // the array must start with "0" as index

 document.write(names[4]); // if the index is out of array length then it displays "UNDEFINED"

</script>

```



## ANOTHER WAY OF CREATING ARRAYS:

```

<script>

 var names = new Array(3);

 names[0] = "Yunus";
 names[1] = "Irshad";
 names[2] = "Zakira";

 document.write(names[1]);

</script>

```



## ARRAY PROPERTIES AND METHODS:

```

<script>

 var names = new Array("Yunus","Irshad","Zakira","Banu");

 document.write(names.length); // length is the property for an array

</script>

```



## CONCAT():

```

<script> // For methods in array you need two arrays

 var names = new Array("Yunus","Irshad","Zakira","Banu");

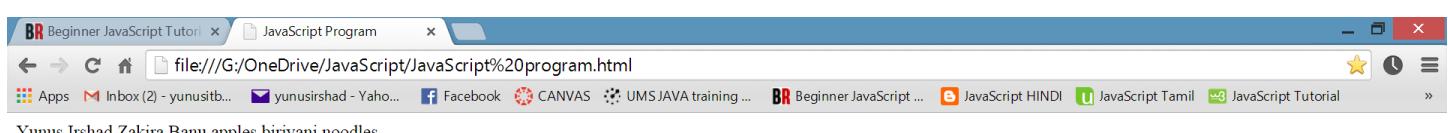
 var food = new Array("apples","biriyani","noodles");

 var concat = names.concat(food); // two arrays add together

 document.write(concat+"
");

</script>

```



## JOIN();

```

<script>

 var names = new Array("Yunus","Irshad","Zakira","Banu");

```

```

var strings = names.join(); // converts the array into strings

document.write(strings+"
");

var strings = names.join(" - "); // it gets separated by " - " hyphen

document.write(strings);

</script>

```

Yunus,Irshad,Zakira,Banu  
Yunus - Irshad - Zakira - Banu

**POP();**

```

<script>

var names = new Array("Yunus","Irshad","Zakira","Banu");

var strings = names.pop(); // pop() method removes the last item from the array

document.write(strings+"
");

</script>

```

Banu

**REVERSE() – PUSH() – SORT();**

```

<script>

var names = new Array("Yunus","Irshad","Zakira","Banu");

document.write(names.reverse()+"
"); // reverses the array items

document.write(names.push("SRK","Salman")+"
"); // adds items into the array

document.write(names.sort()+"
"); // sorts the array items from A to Z

</script>

```

Banu,Zakira,Irshad,Yunus  
6  
Banu,Irshad,SRK,Salman,Yunus,Zakira

## ADDING ARRAY ELEMENTS VIA LOOP USING PROMPT:

```

<script>

var names = new Array(3);

for(var i=0; i<names.length; i++) // filling the array using for loop

```

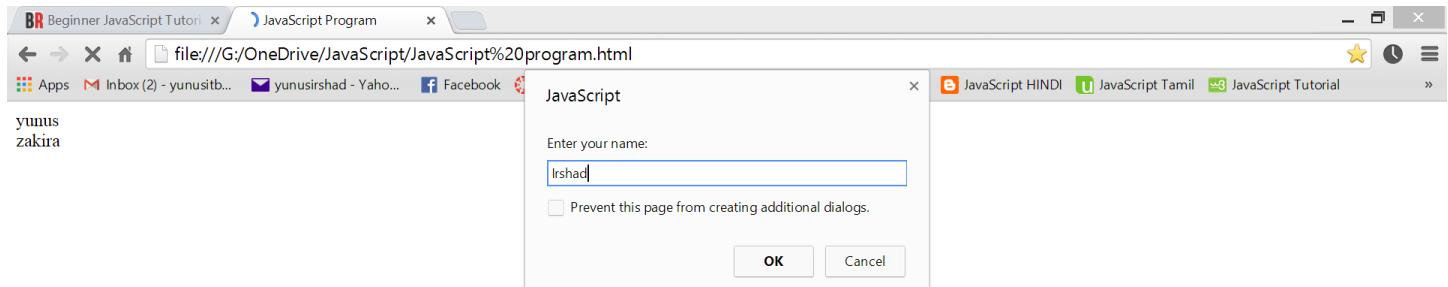
```

{
 names[i] = prompt("Enter your name:","");
// prompting the user to enter the name , the second parameter can be left empty or it will display "Enter
text here"

 document.write(names[i]+
); // displaying the array elements one by one
}

</script>

```



## ASSOCIATIVE ARRAYS:

```

<script>

 var names = new Array();

 names["color"] = "blue"; // instead of indexes we can use string

 names["food"] = "noodles";

 document.write(names["food"]);

</script>

```



## MATH OBJECTS:

```

<script>

 document.write(Math.PI+
); // Pi math object

 document.write(Math.E); // Euler's concept math object

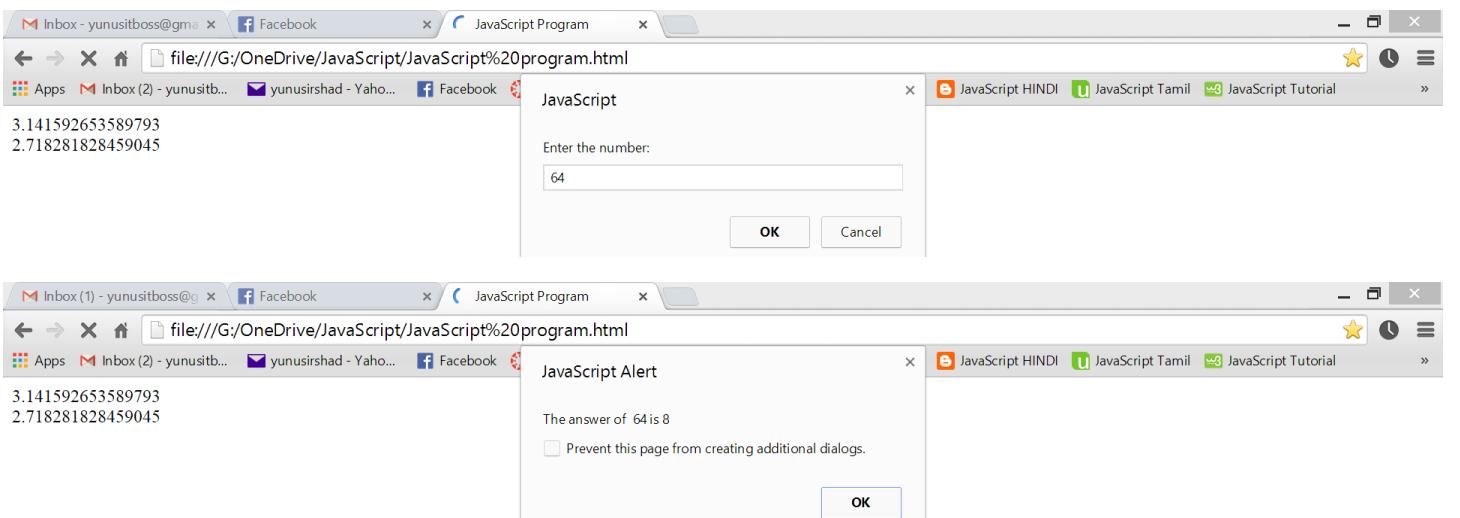
 var a = prompt("Enter the number: "," ");

 var answer = Math.sqrt(a); // square root math object

 alert("The answer of "+a+" is "+answer);

</script>

```

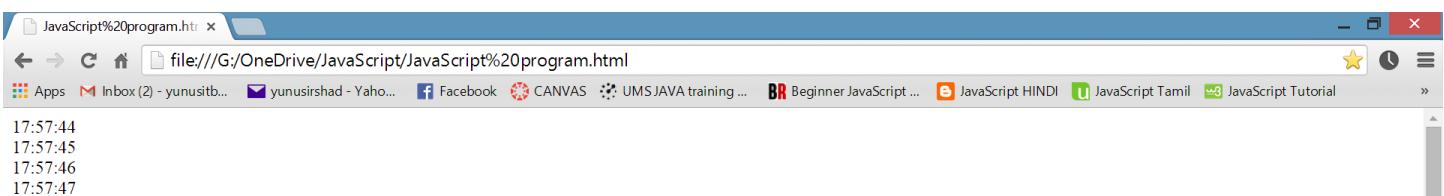


## DATE OBJECTS:

```
<script>

 function printTime()
 {
 var now = new Date(); // Date object
 var hour = now.getHours(); // get hours from the object
 var min = now.getMinutes(); // get minutes from the object
 var sec = now.getSeconds(); // get seconds from the object
 document.write(hour+":"+min+":"+sec+"
");

 }
 setInterval("printTime()",1000); // flashes with an interval of 1000 milliseconds(1sec)
</script>
```



## ACCESSING FORMS:

```
<form> <!--basic form creation-->
 UserName: <input type="text" name="username">
 PassWord: <input type="password" name="password">
 <input type="submit" value="SUBMIT">
</form>
```

```

<script>

 var x = document.forms[0].length;
 // displays the length of elements in forms[0] it is the first form in the program
 document.write(x+"

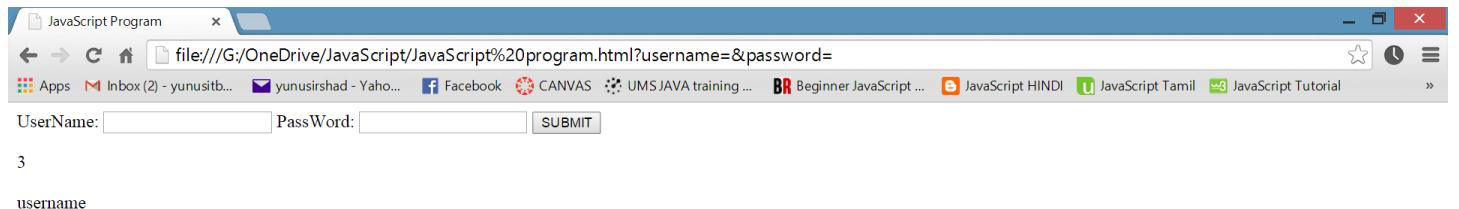
");

 // it displays as 3 because form contains username, password and button

 var y = document.forms[0].elements[0].name;
 // it access the elements[0] in the array and display the declared name
 document.write(y);

</script>

```



## SIMPLE FORM VALIDATION:

```

<form name= "NewForm"> <!--form name is created-->

 <input type="checkbox" name="check">

 <input type="submit" value="CHECK" onclick="validator()">

 <!--javascript which validates the checkbox-->

</form>

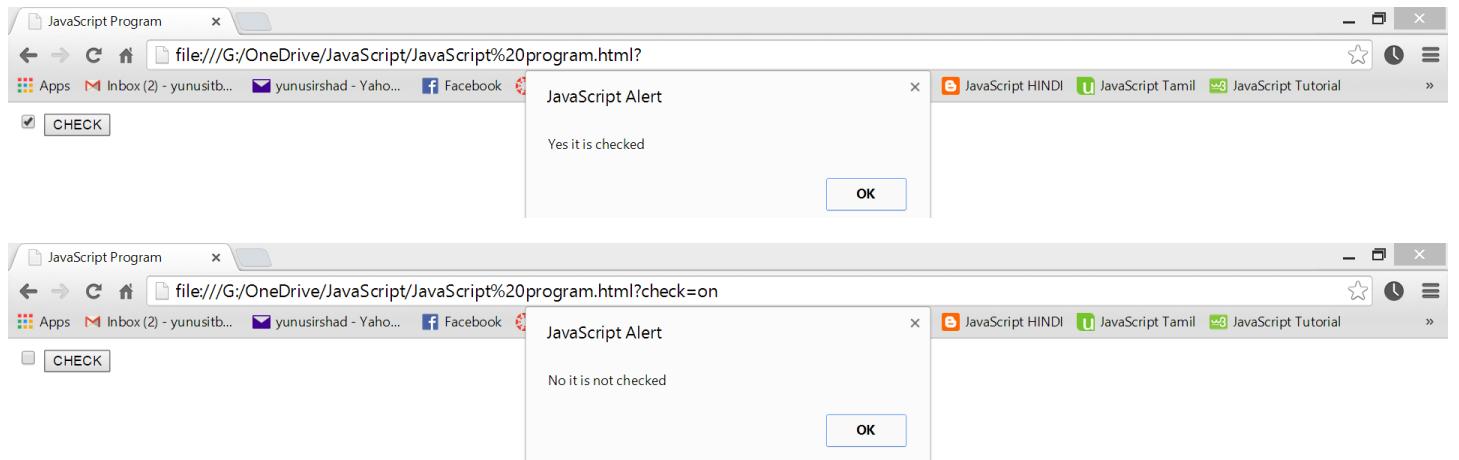
<script>

 function validator()
 {
 if(document.NewForm.check.checked)
 // if statement for checking formname.elementname.checked or not
 {
 alert("Yes it is checked");
 }
 else
 {
 alert("No it is not checked");
 }
 }

```

```
}
```

```
</script>
```



# Software Development Life Cycle Models- Comparison, Consequences

Vanshika Rastogi

Asst. Professor, Dept. of ISE, MVJCE  
Bangalore

**Abstract-** Software Development Life Cycle is a well defined and systematic approach, practiced for the development of a reliable high quality software system. There are tons of SDLC models available. This paper deals with five of those SDLC models, namely; Waterfall model, Iterative model, V-shaped model, Spiral model, agile model. Each development model has certain advantages and disadvantages. The paper begins with the discussion to the introduction of SDLC, followed by the comprehensive comparison among the various SDLC models.

**Key words:** *software development life cycle, development models, comparison between models.*

## 1. INTRODUCTION

The process of building computer software and information systems has been always dictated by different development methodologies. A software development methodology refers to the framework that is used to plan, manage, and control the process of developing an information system. [1] Software Engineering (SE) is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software because it integrates significant mathematics, computer science and practices whose origins are in Engineering. Various processes and methodologies have been developed over the last few decades to improve software quality, with varying degrees of success. However it is widely agreed that no single approach that will prevent project over runs and failures in all cases. Software projects that are large, complicated, poorly-specified, and involve unfamiliar aspects, are still particularly vulnerable to large, unanticipated problems. A software development process is a structure imposed on the development of a software product. There are several models for such processes, each describing approach test to a variety of tasks or activities that take place during the process. It aims to be the standard that defines all the tasks required for developing and maintaining software. [2]

These classic software life cycle models usually include some version or subset of the following activities:

- Planning and Visualization
- Requirement Analysis
- Software Modeling and Design
- Coding
- Documentation
- Testing
- Deployment and Maintenance

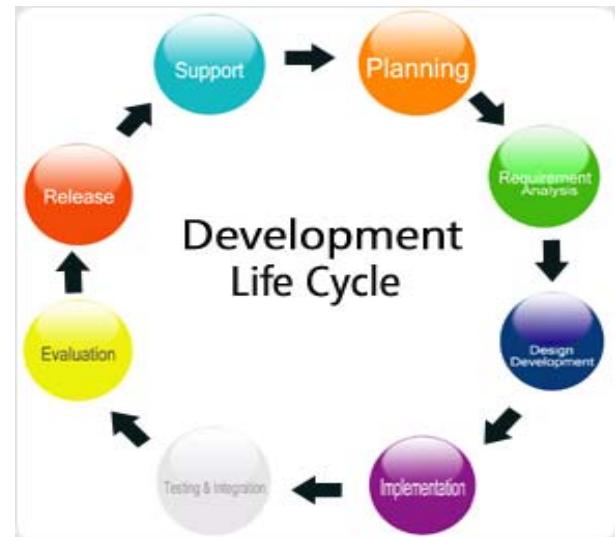


fig.1: SDLC

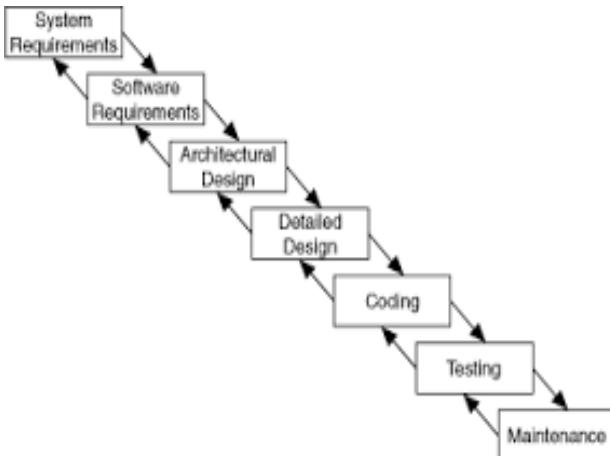
## 2. VARIOUS SDLC MODELS

The various SDLC models are discussed as:

a) **Waterfall model:** The waterfall model is the classical model of software engineering. This model is one of the oldest models and is widely used in government projects and in many major companies. As this model emphasizes planning in early stages, it ensures design flaws before they develop. In addition, it is intensive document and planning makes it work well for projects in which quality control is a major concern. The waterfall life cycle consists of several non overlapping stages; the model begins with establishing system requirements and software requirements and continues with architectural design, detailed design, coding, testing, and maintenance. The waterfall model serves as a baseline for many other life cycle models.[3]

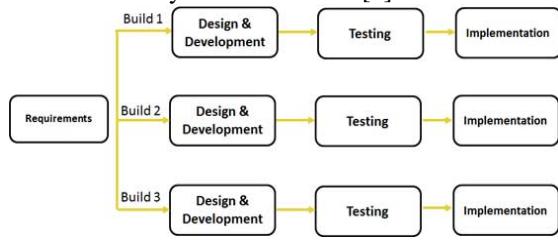
### Basic principle:

- Project is divided into sequential phases, with some overlap and splash back acceptable between phases.
- Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.
- Tight control is maintained over the life of the project via extensive written documentation, formal reviews, and approval/signoff by the user and information technology management to occurring at the end of most phases before beginning the next phase.[2]



**Fig 2: waterfall model**

**b) Iterative model:** An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.[4]



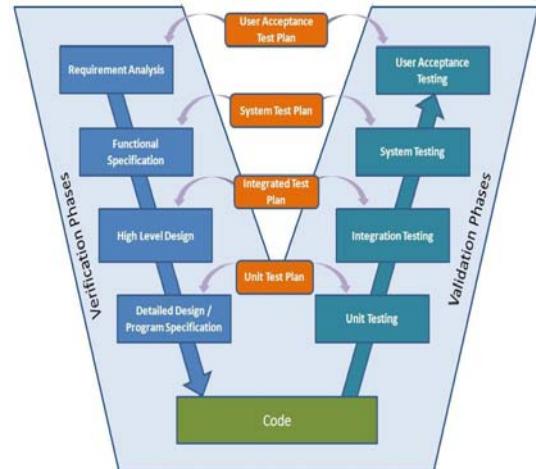
**Fig3: iterative model [5]**

#### **Basic Principle:**

- The problems with the Waterfall Model created a demand for a new method of developing systems which could provide faster results, require less up front information and offer greater flexibility.
- Iterative model, the project is divided into small parts. This allows the development team to demonstrate results earlier on in the process and obtain valuable feedback from system users.
- Each iteration is actually a mini-Waterfall process with the feedback from one phase providing vital Information for the design of the next phase.[2]

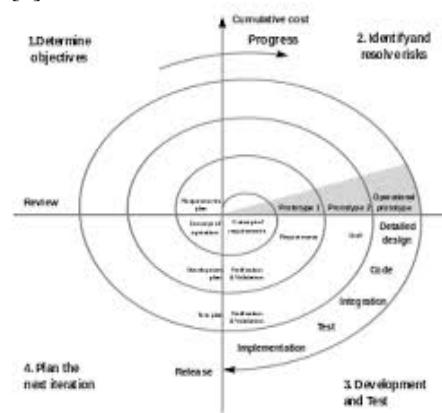
**c) c) V-shaped Model:** Just like the waterfall model, the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. Testing is emphasized in this model more so than the waterfall model though. The testing procedures are developed early in the life cycle before any coding is done, during each of the phases preceding implementation. Requirements begin the life cycle model just like the waterfall model. Before development is started, a system test plan is created. The test plan focuses on meeting the functionality specified in the requirements gathering. The high- level design phase focuses on system architecture and design. An integration test plan is created in this phase as well in order to test the pieces of the

software systems ability to work together. The low- level design phase is where the actual software components are designed, and unit tests are created in this phase as well. The implementation phase is, again, where all coding takes place. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use. [6]



**Fig4: iterative model**

**d) Spiral model:** This model was not the first model [7] to discuss iterative development, but it was the first model to explain why the iteration matters. As originally envisioned, the iterations were typically 6months to 2years long. Each phase starts with a design goal and ends with the client (who may be internal) reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project. The process begins at the center position. From there it moves clock wise in traversals. Each traversal of the spiral usually results in a deliverable [8]. It is not clearly defined what this deliverable is. This changes from traversal to traversal. For example, the first traversals may result in a requirement specification. The second will result in a prototype, and the next one will result in another prototype or sample of a product, until the last traversal leads to a product which is suitable to be sold. Consequently the related activities and their documentation will also mature towards the outer traversals. E.g. a formal design and testing session would be placed into the last traversal.[9]



**Fig5: spiral model**

e) **Agile model:** Agile software development is a style of software development that emphasizes customer satisfaction through continuous delivery of functional software". The Process of Agile Software Development involves the following:

- Starts with a kick-off meeting
- The known requirements are understood and prioritized. The development plan is drawn accordingly.
- Relative complexity of each requirement is estimated
- Sufficient design using simple diagrams is done
- Test Driven Development (TDD) approach may be used. TDD emphasizes on "writing test first and then writing code to pass the test". It can help in avoiding over-coding.
- Development is done, sometimes in pairs, with lot of team interaction. Ownership of code is shared when pair programming is done.
- The code is tested more frequently. Sometime a dedicated "Continuous Integration" Server/Software may be used to ease the integration testing of the code.
- Depending on the feedback received, the code is refactored. Refactoring does not impact the external behavior of the application but the internal structure may be changed to provide better design, maintainability. Some ways of refactoring may be add interface, use super class, move the class etc. [10]

### 3. COMPARISON

The comparison between different models is shown by their advantages and disadvantages in two different tables respectively:

**Table1: Comparison of advantages [6]**

S.N	Waterfall model	Iterative model	Spiral model	V shaped Model
1.	Simple and easy to use.	More flexible than the basic waterfall model.	High amount of risk analysis	Simple and easy to use.
2.	Easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.	If there is personnel continuity between the phases, documentation can be substantially reduced.	Good for large and mission-critical projects	Each phase has specific deliverables.
3.	Phases are processed and completed one at a time.	Implementation of easy areas does not need to wait for the hard ones.	Software is produced early in the software life cycle.	Higher chance of success over the waterfall model due to the development of test plans early on during the life cycle.
4.	Works well for smaller projects where requirements are very well understood.	Works well for smaller and moderate size projects	Works well for projects where risk analysis contains higher priority.	Works well for small projects where requirements are easily understood

**Table2: Comparison of Disadvantages [6]**

S.N	Waterfall model	Iterative model	Spiral model	V shaped Model
1.	Adjusting scope during the life cycle can kill a project.	Milestones are more ambiguous than the waterfall.	Can be a costly model to use.	Very rigid, like the waterfall model.
2.	No working software is produced until late during the life cycle.	Activities performed in parallel are subject to miscommunication and mistaken assumptions.	Risk analysis requires highly specific expertise.	Little flexibility and adjusting scope is difficult and expensive.
3.	High amounts of risk and uncertainty.	Unforeseen interdependencies can create problems.	Project's success is highly dependent on the risk analysis phase.	Software is developed during the implementation phase, so no early prototypes of the software are produced.
4.	Poor model for complex and object-oriented projects. Poor model where requirements are at a moderate to high risk of changing.	Changes are possible as it is iterative model	Doesn't work well for smaller projects.	Model doesn't provide a clear path for problems found during testing phases.

### 4. COMPARISON BETWEEN WATERFALL MODEL AND AGILE MODEL

**Table3: Comparison of Waterfall and Agile model [11]**

	Waterfall model	Agile Model
History	Waterfall model in software engineering got formally introduced as an idea, through a paper published by Winston Royce in 1970.	Agile model of software development, evolved in the 1990s. After further improvements, in 2001, a group of pioneers in agile software development came together and declared the 'Agile Manifesto', which is a set of canonical rules of sorts, for agile software development methods.
Conceptual Difference	It is a sequential process of software development. Just like in a waterfall, the water progressively falls from one altitude to the lower, in a similar way, the production cycle progresses sequentially, from one stage to the other.	Agile breed of models, focus on 'agility' and 'adaptability' in development. Instead of one time-consuming and rigid development schedule, agile models involve multiple iterative development schedules that seek to improve the output with each iteration.
Efficiency	The 'One Phase' and 'Rigid' development cycle of a waterfall model, makes it difficult to make last minute changes in requirements or design.	Agile methods, due to their iterative and adaptable nature, can incorporate changes and release a product, in lesser time.
Suitability	Waterfall model is a natural choice when the customer has provided a clear list of requirements, which are not likely to be modified.	Agile models are applicable in every area of software development. It's best suited for web based applications where its iterative nature helps in incorporating and correcting the various bugs that arise over time.

### 5. CONCLUSION

There are more than tons of sldc models today. Here only a study of five of those models is given. This paper focused on basic five models; their advantages, disadvantages, so that one can select the best suited model as per his requirements.

### 6. FUTURE WORK:

This paper focused on the existing models. There are various shortcomings in the existing models; in future we can have models that can overcome the drawbacks of the existing models.

**REFERENCES:**

- [1] Ian Sommerville, Software Engineering, Addison Wesley, 9th ed., 2010.
- [2] Comparative Analysis of Different types of Models in Software Development Life Cycle-Ms. Shikha Maheshwari, Prof. Dinesh Ch. Jain
- [3] A Comparison between Five Models of Software Engineering Nabil Mohammed Ali Munassar and A.Govardhan
- [4] <http://istqbexamcertification.com/what-is-iterative-model-advantages-disadvantages-and-when-to-use-it/>
- [5] [http://www.tutorialspoint.com/sdlc/sdlc\\_iterative\\_mod\\_el.html](http://www.tutorialspoint.com/sdlc/sdlc_iterative_mod_el.html)
- [6] Raymond Lewallen, "Software Development Life Cycle", 2005
- [7] [weblog.erenkrantz.com/~jerenk/phase-ii/Boe88.pdf](http://weblog.erenkrantz.com/~jerenk/phase-ii/Boe88.pdf)
- [8] [www.ccs.neu.edu/home/matthias/670s05/Lectures/2.html](http://www.ccs.neu.edu/home/matthias/670s05/Lectures/2.html)
- [9] Software Engineering Models Consequences And Alternatives Nitin Mishra, Shantanu Chowdhary, Arunendra Singh, Anil Sharma
- [10] Study & Comparison Of Software Development Lifecycle Models Gourav Khurana, Sachin Gupta
- [11] Gray Pilgrim, "Waterfall Model Vs Agile", Website <http://www.buzzle.com/articles/waterfall-model-vs-agile.html>, Jan, 2012



# SOFTWARE TESTING

**Testing** the software is the process of validating and verifying of a software program.

The errors are to be identified in order to fix those errors. Thus the main objective of software testing is to maintain and deliver a quality product to the client.

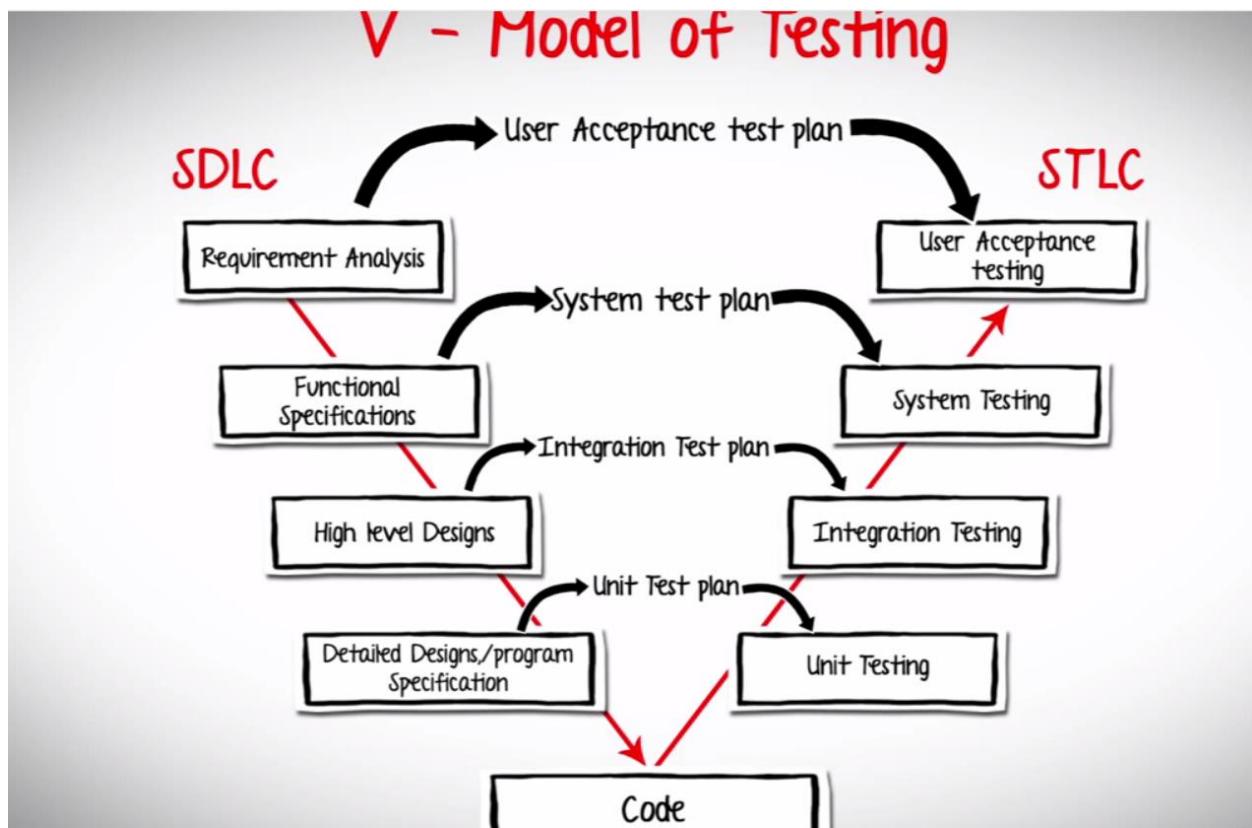
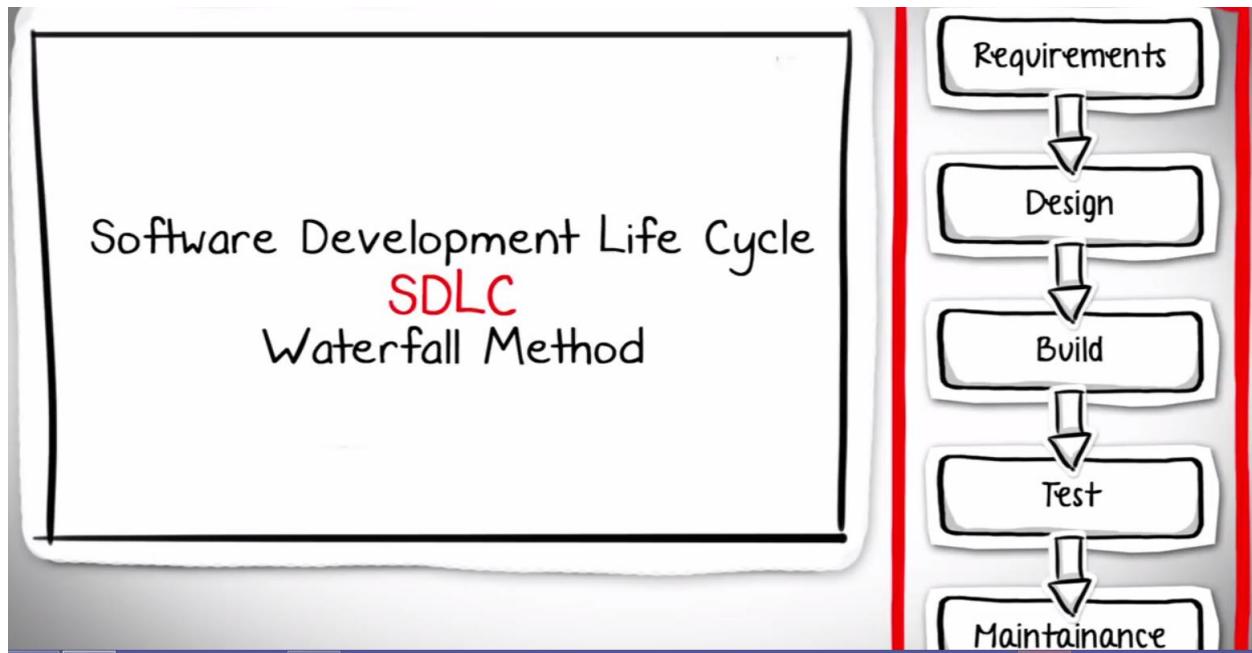
A banking software is entirely different from a software required in a shop. The needs and requirements of both those software are different. Hence it is important to check its potential.

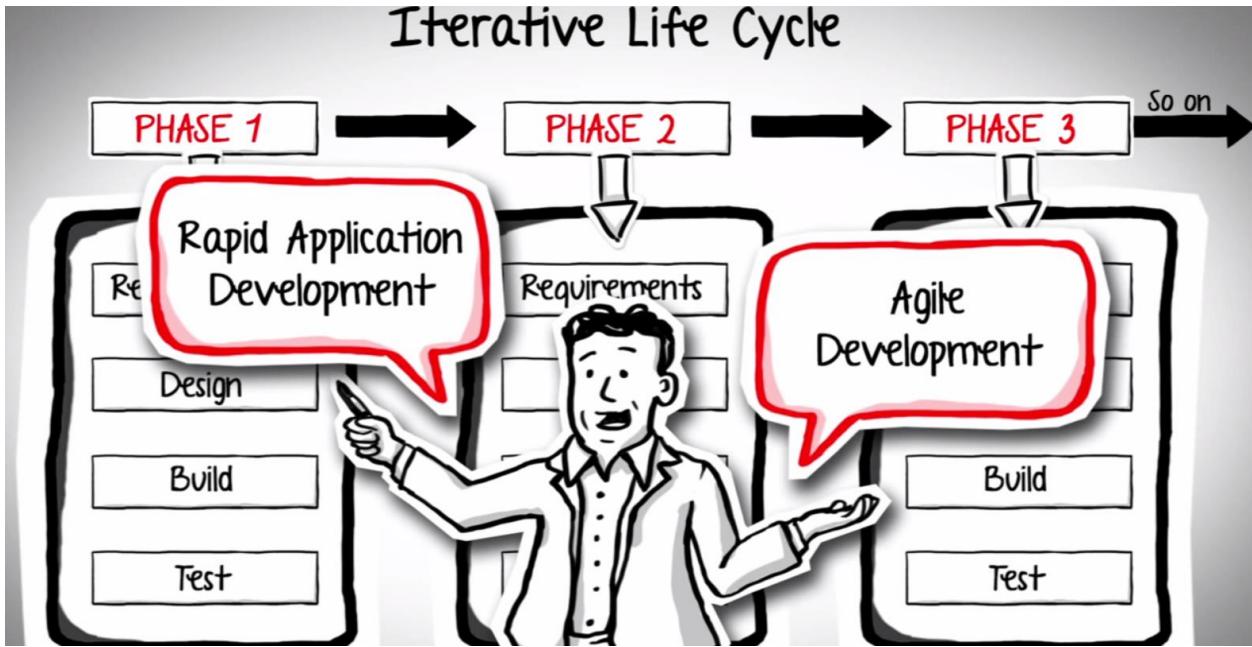
The main goal of software testing is to know the errors of the software before the user finds them.

Software testing helps to give a quality certification that the software can be used by the client immediately

- For e.g.: Airbus crashed 250 people died... radio therapy 3 died extra doses... satellite launch was crashed .....
- Microsoft windows 98 crashes on the launch day ....

7 Testing Principles	
Principle 1	Testing shows presence of defects
Principle 2	Exhaustive testing is impossible
Principle 3	Early Testing
Principle 4	Defect Clustering
Principle 5	Pesticide Paradox
Principle 6	Testing is context dependent
Principle 7	Absence of errors - fallacy





**Testing can done in two ways:**

1. *Positive software testing*
2. *Negative software testing*

Positive software testing is the testing by giving the expected data to know whether it works well with expected data, whereas negative software testing is testing by giving what is unexpected to know how the software reacts to such a data.

## Types of Software Testing

We have come across so many **types of software testing**. The two major approaches of software testing are *manual software testing and automated software testing*. Manual software testing means it is being done by a man. That means a person, i.e. a tester runs the software for errors. The following are the stages:

1. Unit testing
2. Integration testing
3. System testing
4. User acceptance testing

Different techniques are used for software techniques. The commonly used **software testing techniques** are the following:

- White box testing
- Black box testing
- Grey box testing

## White box testing:

White box testing is the detailed investigation of internal logic and structure of the code. White box testing is also called glass testing or open box testing. In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code.

The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

Advantages	Disadvantages
<p>As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively.</p> <p>It helps in optimizing the code.</p> <p>Extra lines of code can be removed which can bring in hidden defects.</p> <p>Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing.</p>	<p>Due to the fact that a skilled tester is needed to perform white box testing, the costs are increased.</p> <p>Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems as many paths will go untested.</p> <p>It is difficult to maintain white box testing as the use of specialized tools like code analyzers and debugging tools are required.</p>

**Black box testing:** The technique of testing without having any knowledge of the interior workings of the application is Black Box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

Advantages	Disadvantages
<p>Well suited and efficient for large code segments.</p> <p>Code Access not required.</p> <p>Clearly separates user's perspective from the developer's perspective through visibly defined roles.</p> <p>Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language or operating systems.</p>	<p>Limited Coverage since only a selected number of test scenarios are actually performed.</p> <p>Inefficient testing, due to the fact that the tester only has limited knowledge about an application.</p> <p>Blind Coverage, since the tester cannot target specific code segments or error prone areas.</p>

	The test cases are difficult to design.
--	-----------------------------------------

**Grey box software testing** is the combination of white box as well as black box testing.

Grey Box testing is a technique to test the application with limited knowledge of the internal workings of an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black box testing, where the tester only tests the application's user interface, in grey box testing, the tester has access to design documents and the database. Having this knowledge, the tester is able to better prepare test data and test scenarios when making the test plan.

Advantages	Disadvantages
<p>Offers combined benefits of black box and white box testing wherever possible.</p> <p>Grey box testers don't rely on the source code; instead they rely on interface definition and functional specifications.</p> <p>Based on the limited information available, a grey box tester can design excellent test scenarios especially around communication protocols and data type handling.</p> <p>The test is done from the point of view of the user and not the designer.</p>	<p>Since the access to source code is not available, the ability to go over the code and test coverage is limited.</p> <p>The tests can be redundant if the software designer has already run a test case.</p> <p>Testing every possible input stream is unrealistic because it would take an unreasonable amount of time; therefore, many program</p>

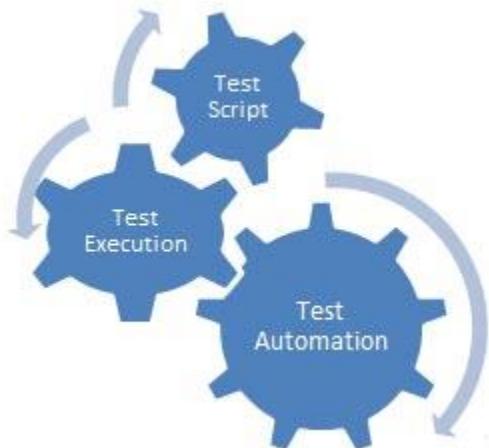
### Black Box vs Grey Box vs White Box

S.N.	Black Box Testing	Grey Box Testing	White Box Testing
1	The Internal Workings of an application are not required to be known	Somewhat knowledge of the internal workings are known	Tester has full knowledge of the Internal workings of the application
2	Also known as closed box testing, data driven	Another term for grey box testing is translucent testing	Also known as clear box testing, structural

	testing and functional testing	as the tester has limited knowledge of the insides of the application	testing or code based testing
3	Performed by end users and also by testers and developers	Performed by end users and also by testers and developers	Normally done by testers and developers
4	Testing is based on external expectations - Internal behavior of the application is unknown	Testing is done on the basis of high level database diagrams and data flow diagrams	Internal workings are fully known and the tester can design test data accordingly
5	This is the least time consuming and exhaustive	Partly time consuming and exhaustive	The most exhaustive and time consuming type of testing
6	Not suited to algorithm testing	Not suited to algorithm testing	Suited for algorithm testing
7	This can only be done by trial and error method	Data domains and Internal boundaries can be tested, if known	Data domains and Internal boundaries can be better tested

## Automation testing

Automation testing which is also known as *Test Automation*, is when the tester writes scripts and uses another software to test the software. This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly and repeatedly.



Apart from regression testing, Automation testing is also used to test the application from load, performance and stress point of view. It increases the test coverage; improve accuracy, saves time and money in comparison to manual testing.



## Comparison of Frameworks

Approach	Advantages	Disadvantages
Modular testing framework	Modular approach Reusable functions Hierarchical Structure	Test data within the scripts limits reusability, Test script is dependent on software.
Data driven testing framework	Improved Maintainability	Dependency on technical expertise, Test script is dependent on software.
Keyword driven testing framework	Ease of maintenance, Scalability, Less dependency of software.	Dependency on technical expertise, Requires large effort
Hybrid testing framework	Integrates the advantages of all the other frameworks.	Increased Complexity

### **What to automate?**

It is not possible to automate everything in the Software; however the areas at which user can make transactions such as login form or registration forms etc, any area where large amount of users. Can access the Software simultaneously should be automated.

Furthermore all GUI items, connections with databases, field validations etc can be efficiently tested by automating the manual process.

### **When to automate?**

Test Automation should be used by considering the following for the Software:

- Large and critical projects.
- Projects that require testing the same areas frequently.
- Requirements not changing frequently.
- Accessing the application for load and performance with many virtual users.
- Stable Software with respect to manual testing.
- Availability of time.

### **How to automate?**

Automation is done by using a supportive computer language like vb scripting and an automated software application. There are a lot of tools available which can be used to write automation scripts. Before mentioning the tools let's identify the process which can be used to automate the testing:

- Identifying areas within a software for automation.
- Selection of appropriate tool for Test automation.
- Writing Test scripts.
- Development of Test suits.
- Execution of scripts.
- Create result reports.
- Identify any potential bug or performance issue.
- Tools used IBM Rational Robot , Silk performer and QTP
- **Performance Testing:** Tools used WEB Load, Load runner.

- ✓ **SCOPE OF AUTOMATION NEEDS TO BE DETERMINED IN DETAIL BEFORE THE START OF THE PROJECT.**  
**THIS SETS EXPECTATIONS FROM AUTOMATION RIGHT.**
- ✓ **SELECT THE RIGHT AUTOMATION TOOL:**  
**A TOOL MUST NOT BE SELECTED BASED ON ITS POPULARITY BUT IT'S FIT TO THE AUTOMATION REQUIREMENTS.**
- ✓ **CHOOSE APPROPRIATE FRAMEWORK**

•

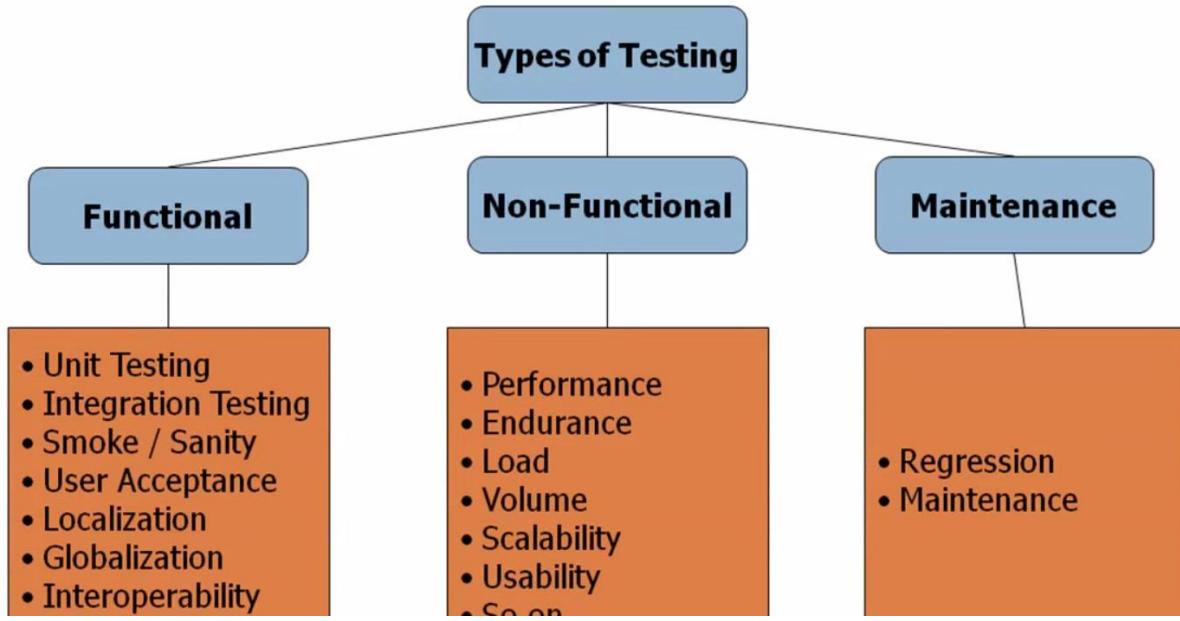
•

## Software testing tools

Following are the tools which can be used for Automation testing:

- HP Quick Test Professional
- Selenium
- IBM Rational Functional Tester
- SilkTest
- TestComplete
- Testing Anywhere
- WinRunner
- LaodRunner
- Visual Studio Test Professional
- WATIR

## Levels of Software Testing



The main levels of Software Testing:

- Functional Testing.
- Non-Functional Testing.

- Apart from Functional testing, **non-functional requirements like performance, usability, load factor** are also important
- **Performance Testing** : Check & Fine tune system response time. The goal here is to reduce the response time
- **Load Testing** : System performance at different load i.e. number of people accessing the system



**Server Busy**

The request cannot be processed at this time.

## Functional Testing

This is a type of black box testing that is based on the specifications of the software that is to be tested

### **Unit Testing (Component testing)**

This type of testing is performed by the developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is separate from the test data of the quality assurance team.

The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

### **LIMITATIONS OF UNIT TESTING**

Testing cannot catch each and every bug in an application. It is impossible to evaluate every execution path in every software application. The same is the case with unit testing.



### **Integration Testing**

The testing of combined parts of an application to determine if they function correctly together is Integration testing. There are two methods of doing Integration Testing Bottom-up Integration testing and Top down Integration testing.

S.N.	Integration Testing Method
1	<b>Bottom-up integration</b> This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.
2	<b>Top-Down integration</b> This testing, the highest-level modules are tested first and progressively lower-level modules are tested after that.

## System Testing

This is the next level in the testing and tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets Quality Standards. This type of testing is performed by a specialized testing team.

### Retest:

When a bug is fixed by the developer, testing the same bug to ensure whether it has been fixed or not is known as retesting.

### Statement Coverage in software testing?

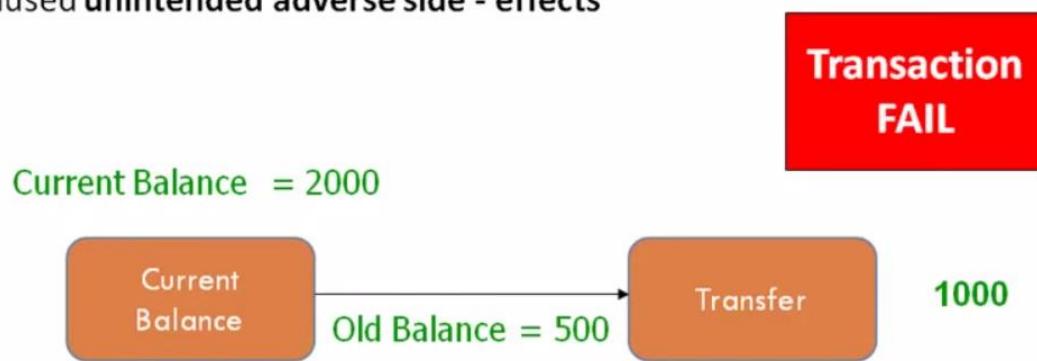
Statement coverage is one of the ways of measuring code coverage. It describes the degree to which the software code of a program has been tested.

All the statements in the code must be executed and tested.

## Regression Testing

Whereas testing the other features of the application which might be affected by the bug fix is known as **regression testing**. The intent of Regression testing is to ensure that a change, such as a bug fix did not result in another fault being uncovered in the application.

- As you may observe , **code changes were in Current Balance module only** but still transfer module is effected.
- Regression testing is carried out to check modification in software has not caused **unintended adverse side - effects**



## Acceptance Testing

This is arguably the most importance type of testing as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client.s requirements. The QA team will have a set of pre written scenarios and Test Cases that will be used to test the application.

Acceptance tests are not only intended to point out simple spelling mistakes, cosmetic errors or Interface gaps, but also to point out any bugs in the application that will result in system crashers or major errors in the application.

## **ALPHA TESTING**

This test is the first stage of testing and will be performed amongst the teams (developer and QA teams). Unit testing, integration testing and system testing when combined are known as alpha testing. During this phase, the following will be tested in the application:

- Spelling Mistakes
- Broken Links
- Cloudy Directions
- The Application will be tested on machines with the lowest specification to test loading times and any latency problems.

## **BETA TESTING**

This test is performed after Alpha testing has been successfully performed. In beta testing a sample of the intended audience tests the application. Beta testing is also known as pre-release testing. Beta test versions of software are ideally distributed to a wide audience on the Web, partly to give the program a "real-world" test and partly to provide a preview of the next release. In this phase the audience will be testing the following:

- Users will install, run the application and send their feedback to the project team.
- Typographical errors, confusing application flow, and even crashes.
- Getting the feedback, the project team can fix the problems before releasing the software to the actual users.
- The more issues you fix that solve real user problems, the higher the quality of your application will be.

## **Performance Testing**

It is mostly used to identify any bottlenecks or performance issues rather than finding the bugs in software. There are different causes which contribute in lowering the performance of software:

- Network delay.
- Client side processing.
- Database transaction processing.

- Load balancing between servers.
- Data rendering.

Performance testing is considered as one of the important and mandatory testing type in terms of following aspects:

- Speed (i.e. Response Time, data rendering and accessing)
- Capacity
- Stability
- Scalability

It can be either qualitative or quantitative testing activity and can be divided into different sub types such as **Load testing** and **Stress testing**.

## **LOAD TESTING**

A process of testing the behavior of the Software by applying maximum load in terms of Software accessing and manipulating large input data. It can be done at both normal and peak load conditions. This type of testing identifies the maximum capacity of Software and its behavior at peak time.

Most of the time, Load testing is performed with the help of automated tools such as Load Runner, AppLoader, IBM Rational Performance Tester, Apache JMeter, Silk Performer, Visual Studio Load Test etc.

Virtual users (VUsers) are defined in the automated testing tool and the script is executed to verify the Load testing for the Software. The quantity of users can be increased or decreased concurrently or incrementally based upon the requirements.

## **STRESS TESTING**

This testing type includes the testing of Software behavior under abnormal conditions. Taking away the resources, applying load beyond the actual load limit is Stress testing.

The main intent is to test the Software by applying the load to the system and taking over the resources used by the Software to identify the breaking point. This testing can be performed by testing different scenarios such as:

- Shutdown or restart of Network ports randomly.
- Turning the database on or off.
- Running different processes that consume resources such as CPU, Memory, server etc.

## Usability Testing

This section includes different concepts and definitions of Usability testing from Software point of view. It is a black box technique and is used to identify any error(s) and improvements in the Software by observing the users through their usage and operation.

According to Nielsen, Usability can be defined in terms of five factors i.e. Efficiency of use, Learnability, Memorability, Errors/safety, satisfaction. According to him the usability of the product will be good and the system is usable if it possesses the above factors.

## UI VS USABILITY TESTING

UI testing involves the testing of Graphical User Interface of the Software. This testing ensures that the GUI should be according to requirements in terms of color, alignment, size and other properties.

On the other hand Usability testing ensures that a good and user friendly GUI is designed and is easy to use for the end user. UI testing can be considered as a sub part of Usability testing.

## Security Testing

Security testing involves the testing of Software in order to identify any flaws and gaps from security and vulnerability point of view. Following are the main aspects which Security testing should ensure:

- Confidentiality.
- Integrity.
- Authentication.
- Availability.
- Authorization.
- Non-repudiation.
- Software is secure against known and unknown vulnerabilities.
- Software data is secure.
- Software is according to all security regulations.
- Input checking and validation.
- SQL insertion attacks.
- Injection flaws.

- Session management issues.
- Cross-site scripting attacks.
- Buffer overflows vulnerabilities.
- Directory traversal attacks.

### **Portability Testing**

Portability testing includes the testing of Software with intent that it should be re-useable and can be moved from another Software as well. Following are the strategies that can be used for Portability testing.

- Transferred installed Software from one computer to another.
- Building executable (.exe) to run the Software on different platforms.

Portability testing can be considered as one of the sub parts of System testing, as this testing type includes the overall testing of Software with respect to its usage over different environments. Computer Hardware, Operating Systems and Browsers are the major focus of Portability testing. Following are some pre-conditions for Portability testing:

- Software should be designed and coded, keeping in mind Portability Requirements.
- Unit testing has been performed on the associated components.
- Integration testing has been performed.
- Test environment has been established.

### **Verification & Validation**

S.N.	Verification	Validation
1	Are you building it right?	Are you building the right thing?
2	Ensure that the software system meets all the functionality.	Ensure that functionalities meet the intended behavior.
3	Verification takes place first and includes the checking for documentation, code etc.	Validation occurs after verification and mainly involves the checking of the overall product.
4	Done by developers.	Done by Testers.

5	Have static activities as it includes the reviews, walkthroughs, and inspections to verify that software is correct or not.	Have dynamic activities as it includes executing the software against the requirements.
6	It is an objective process and no subjective decision should be needed to verify the Software.	It is a subjective process and involves subjective decisions on how well the Software works.

## Software Testing Myths

<b>1 Myths: Testing is too expensive.</b>	<b>Reality:</b> There is a saying, pay less for testing during software development or pay more for maintenance or correction later. Early testing saves both time and cost in many aspects however, reducing the cost without testing may result in the improper design of a software application rendering the product useless.
<b>2 Myths: Testing is time consuming.</b>	<b>Reality:</b> During the SDLC phases testing is never a time consuming process. However diagnosing and fixing the error which is identified during proper testing is a time consuming but productive activity.
<b>3 Myths: Testing cannot be started if the product is not fully developed.</b>	<b>Reality:</b> No doubt, testing depends on the source code but reviewing requirements and developing test cases is independent from the developed code. However iterative or incremental approach as a development life cycle model may reduce the dependency of testing on the fully developed software.
<b>4 Myths: Complete Testing is Possible.</b>	<b>Reality:</b> It becomes an issue when a client or tester thinks that complete testing is possible. It is possible that all paths have been tested by the team but occurrence of complete testing is never possible. There might be some scenarios that are never executed by the test team or the client during the software development life cycle and may be executed once the project has been deployed.
<b>5 Myths: If the software is tested then it must be bug free.</b>	<b>Reality:</b> This is a very common myth which clients, Project Managers and the management team believe in. No one can say with absolute certainty that a software application is 100% bug free even if a tester with superb testing skills has tested the application.
<b>6 Myths: Missed defects are due to Testers.</b>	

	<b>Reality:</b> It is not a correct approach to blame testers for bugs that remain in the application even after testing has been performed. This myth relates to Time, Cost, and Requirements changing Constraints. However the test strategy may also result in bugs being missed by the testing team.
<b>7</b>	<b>Myths: Testers should be responsible for the quality of a product.</b>  <b>Reality:</b> It is a very common misinterpretation that only testers or the testing team should be responsible for product quality. Tester.s responsibilities include the identification of bugs to the stakeholders and then it is their decision whether they will fix the bug or release the software. Releasing the software at the time puts more pressure on the testers as they will be blamed for any error.
<b>8</b>	<b>Myths: Test Automation should be used wherever it is possible to use it and to reduce time.</b>  <b>Reality:</b> Yes it is true that Test Automation reduces the testing time but it is not possible to start Test Automation at any time during Software development. Test Automaton should be started when the software has been manually tested and is stable to some extent. Moreover, Test Automation can never be used if requirements keep changing.
<b>9</b>	<b>Myths: Any one can test a Software application.</b>  <b>Reality:</b> People outside the IT industry think and even believe that any one can test the software and testing is not a creative job. However testers know very well that this is myth. Thinking alternatives scenarios, try to crash the Software with the intent to explore potential bugs is not possible for the person who developed it.
<b>10</b>	<b>Myths: A tester's task is only to find bugs.</b>  <b>Reality:</b> Finding bugs in the Software is the task of testers but at the same time they are domain experts of the particular software. Developers are only responsible for the specific component or area that is assigned to them but testers understand the overall workings of the software, what the dependencies are and what the impacts of one module on another module are.

## Testing, Quality Assurance and Quality Control

S.N.	Quality Assurance	Quality Control	Testing
1	Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements.	Activities which ensure the verification of developed software with respect to documented (or not in some cases) requirements.	Activities which ensure the identification of bugs/error/defects in the Software.

2	Focuses on processes and procedures rather than conducting actual testing on the system.	Focuses on actual testing by executing Software with intent to identify bug/defect through implementation of procedures and process.	Focuses on actual testing.
3	Process oriented activities.	Product oriented activities.	Product oriented activities.
4	Preventive activities.	It is a corrective process.	It is a preventive process.
5	It is a subset of Software Test Life Cycle (STLC).	QC can be considered as the subset of Quality Assurance.	Testing is the subset of Quality Control.

## Software Testing Documentation

Testing documentation involves the documentation of artifacts which should be developed before or during the testing of Software.

Documentation for Software testing such as:

- Test Plan
- Test Scenario
- Test Case
- Traceability Matrix

### Test Plan

A test plan outlines the strategy that will be used to test an application, the resources that will be used, the test environment in which testing will be performed, the limitations of the testing and the schedule of testing activities. Typically the Quality Assurance Team Lead will be responsible for writing a Test Plan.

A test plan will include the following.

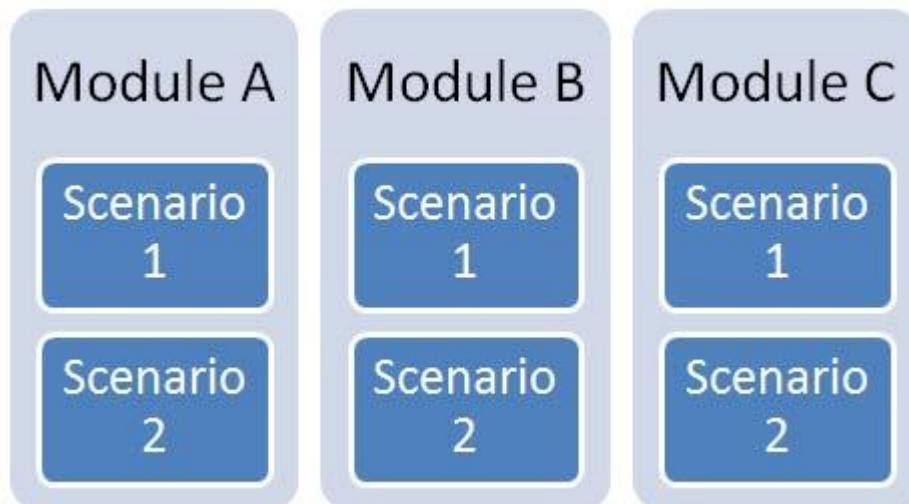
- Introduction to the Test Plan document
- Assumptions when testing the application

- List of test cases included in Testing the application
- List of features to be tested
- What sort of Approach to use when testing the software
- List of Deliverables that need to be tested
- The resources allocated for testing the application
- Any Risks involved during the testing process
- A Schedule of tasks and milestones as testing is started

### **Test Scenario or Test condition or Test possibility**

A one line statement that tells what area in the application will be tested. Test Scenarios are used to ensure that all process flows are tested from end to end. A particular area of an application can have as little as one test scenario to a few hundred scenarios depending on the magnitude and complexity of the application.

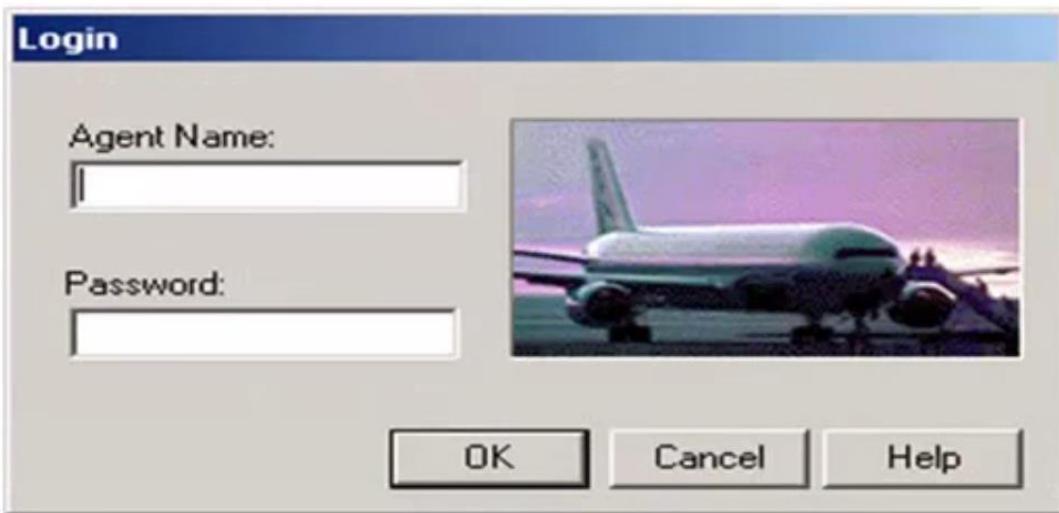
The term test scenario and test cases are used interchangeably however the main difference being that test scenarios has several steps however test cases have a single step. When viewed from this perspective test scenarios are test cases, but they include several test cases and the sequence that they should be executed. Apart from this, each test is dependent on the output from the previous test.



## Scenario 1

**Guru99**

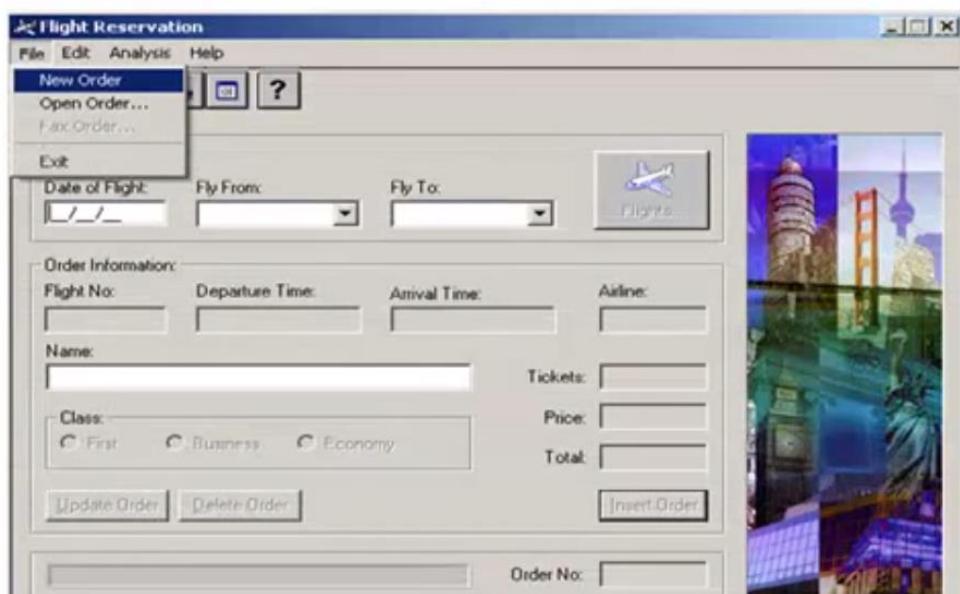
- Check Login Functionality



## Scenario 2

**Software Testing**

- Check New Order



## Exercise

- 1) Check Login
- 2) Check New Order
- 3) Check Open Order
- 4) Check Fax Order
- 5) Check Help
- 6) Check About

## Test Case

Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks. The main intent of this activity is to ensure whether the Software Passes or Fails in terms of its functionality and other aspects. There are many types of test cases like: functional, negative, error, logical test cases, physical test cases, UI test cases etc.

Furthermore test cases are written to keep track of testing coverage of Software. Generally, there is no formal template which is used during the test case writing. However, following are the main components which are always available and included in every test case:

- Test case ID.
- Product Module.
- Product version.
- Revision history.
- Purpose
- Assumptions
- Pre-Conditions.
- Steps.
- Expected Outcome.
- Actual Outcome.
- Post Conditions.

Many Test cases can be derived from a single test scenario. In addition to this, some time it happened that multiple test cases are written for single Software which is collectively known as test suites.

Test Scenario	Test Case	Pre Conditions	Test Step	Test Data	Expected Result	Actual Results	Pass/Fail
Check Login Functionality	Check response on Entering valid Agent Name & Password	Flight Reservation Application must be installed	<ol style="list-style-type: none"> <li>1. Launch Application</li> <li>2. Enter Agent Name</li> <li>3. Enter Password</li> <li>4. Click OK button</li> </ol>	Agent Name : guruv99 Password : MERCURY	Login must be successful.	Login Successful	Pass

### Traceability Matrix

Traceability Matrix (also known as Requirement Traceability Matrix - RTM) is a table which is used to trace the requirements during the Software development life Cycle. It can be used for forward tracing (i.e. from Requirements to Design or Coding) or backward (i.e. from Coding to Requirements). There are many user defined templates for RTM.

Each requirement in the RTM document is linked with its associated test case, so that testing can be done as per the mentioned requirements. Furthermore, Bug ID is also include and linked with its associated requirements and test case. The main goals for this matrix are:

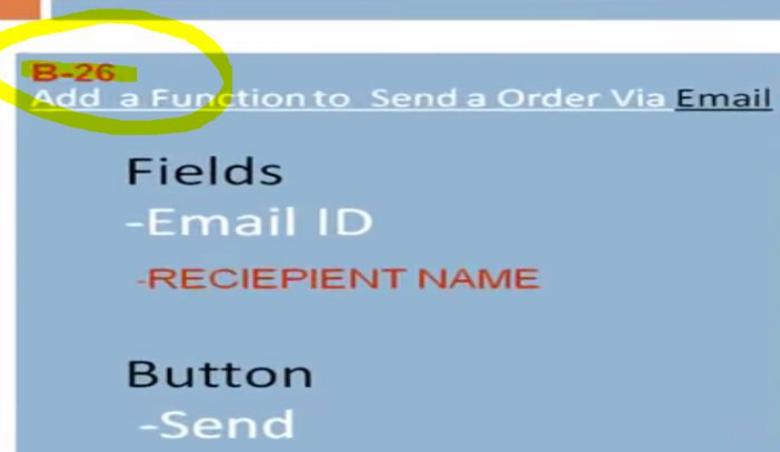
- Make sure Software is developed as per the mentioned requirements.
- Helps in finding the root cause of any bug.
- Helps in tracing the developed documents during different phases of SDLC.

### Change in Requirement

B-26  
Add a Function to Send a Order Via Email

**Fields**  
-Email ID  
-RECIPIENT NAME

**Button**  
-Send



### **Bi-directional Traceability Matrix?**

Bidirectional Traceability refers to the Forward and Backward traceability.

Forward Traceability is from requirements to design to code to test cases.

Whereas the Backward Traceability is in the reverse direction

### **Defect Leakage?**

Defect leakage refers to the defect Found \ reproduced by the Client or User, which the tester was unable to find.

### **Test Bed?**

Test bed is the environment which is required to test the software.

This includes requirement of Hardware, Software, Memory, CPU speed, operating system etc.

### **Latent Bug?**

**Latent Bug** is a bug, which gets unobserved in two or more releases of the application.

### **Base Lining?**

**Base lining** is the process by which the quality and cost effectiveness of a service is assessed, usually in advance of a change to the service.

### **Measure of Completeness in software testing?**

In software testing there are two measures of completeness, code coverage and path coverage.

**Code coverage** is a white box testing technique to determine how much of a program's source code has been tested. There are several fronts on which code coverage is measured. Whereas **Path coverage** establishes whether every potential route through a segment of code has been executed and tested.

### **Pair Programming?**

Pair Programming is a software development approach whereby lines of code of a component are written by two programmers sitting at a single computer

### **N+1 Testing?**

N+1 Testing is a variation of Regression Testing. It involves testing conducted with multiple cycles in which errors found in test cycle 'N' are resolved and the solution is re-tested in test cycle N+1. The cycles are typically repeated until the solution reaches a steady state and there are no errors.

**Difference between an application server and a Web server?**

Web server serves pages for viewing in a Web browser, while an application server provides methods that client applications can call.

**Error Seeding?**

Error Seeding is the process of intentionally adding known defects to those already in the component or system for the purpose of monitoring the rate of detection and removal, and estimating the number of remaining defects.

**Cause Effect Analysis?**

Cause effect analysis is an approach for studying the specifications carefully and identifying the combinations of input conditions or causes and their effect in the form of a table and designing test cases.

**Error Guessing?**

Error guessing is a supplementary technique of test case design involving test case design based on the tester's intuition and experience.

**Basis Path Testing?**

Basis Path Testing is white box testing method involving design of test cases to cover every statement, every branch and every condition in the code which has been written.

**Fuzz Testing?**

Fuzz testing a technique of testing an application by feeding random inputs.

**Failure Mode and Effect Analysis (FMEA)?**

Failure Mode and Effect Analysis is a systematic approach to risk identification and analysis of identifying possible modes of failure and attempting to prevent their occurrence.

**Blocked Test Case?**

Blocked Test Case refers to the test case, which cannot be executed because the preconditions for its execution are not fulfilled.

# SQL LANGUAGE (MySQL)

- 1 - Introduction to Databases
- 2 - Getting a MySQL Server
- 3 - Creating a Database
- 4 - SHOW and SELECT
- 5 - Basic Rules for SQL Statements
- 6 - Getting Multiple Columns
- 7 - DISTINCT and LIMIT
- 8 - Sorting Results
- 9 - Sort Direction
- 10 - Basic Data Filtering and WHERE
- 11 - Advanced Filtering Using AND and OR
- 12 - Are you IN or are you NOT IN?
- 13 - How Search Engines Work
- 14 - More on Wildcards
- 15 - Regular Expressions
- 16 - Creating Custom Columns
- 17 - Functions
- 18 - More on Aggregate Functions
- 19 - GROUP BY
- 20 - Subqueries
- 21 - Another Subquery Example
- 22 - How to Join Tables
- 23 - Outer Joins
- 24 - UNION
- 25 - Full-Text Searching
- 26 - INSERT INTO
- 27 - How to Insert Multiple Rows
- 28 - UPDATE and DELETE
- 29 - CREATE TABLE
- 30 - NOT NULL and AUTO INCREMENT
- 31 - ALTER / DROP / RENAME TABLE
- 32 - Views
- 33 - Final Video!

## SHOW and SELECT COMMAND:

Install 000webhost.com click phpMyDomain....

Click SQL and type coding .... It displays the Tables name

The screenshot shows the 'SQL query' tab in the phpMyAdmin interface. The input field contains the command 'SHOW TABLES'. The results pane is empty.

The screenshot shows the 'Tables\_in\_a5581884\_creatDB' table. It has one row named 'customers'.

It displays the column names from the customers table ....

The screenshot shows the 'SQL query' tab with the command 'SHOW COLUMNS FROM customers'. To the left, there is a tree view showing a single node 'customers'. The results pane displays the column structure of the 'customers' table.

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	
name	varchar(60)	NO		NULL	
address	varchar(60)	NO		NULL	
city	varchar(30)	NO		NULL	
state	varchar(10)	NO		NULL	
zip	int(11)	NO		NULL	

Displays the column details of the customer table so we use SELECT:

The screenshot shows the 'SQL query' tab with the query 'SELECT CITY FROM customers LIMIT 0 , 30'. Below it, there are options to 'Show : 30' and 'in horizontal'. The results pane shows a table titled 'CITY' with the following data:

	CITY
1	Adams
2	Raleigh
3	Oakland
4	Simmersville
5	Newark
6	Gary

If you are executing multiple queries then you must use semicolon at the end of the queries ....

SELECT city FROM customers;

SELECT state FROM customers;

SQL is **not case sensitive** but use only capitals .....

Displaying multiple columns .....

```
SELECT name,city FROM customers
```

Displaying all the columns in the table.....

```
SELECT * FROM customers
```

Displaying the column data which eliminates the duplicate of rows ... just print unique rows which doesn't repeat twice or more....

```
SELECT DISTINCT city FROM customers
```

Displaying the column data with a limit of 5 rows.....

```
SELECT * FROM customers LIMIT 5
```

Displaying the column data with a limit of 10 rows but from different position say from 13<sup>th</sup> row....

Computer reads from 0 so the 13<sup>th</sup> row is taken as from 14<sup>th</sup> row.....

```
SELECT * FROM customers LIMIT 13,10
```

Sort by key:   None							
«<T>>	id	name	address	city	state	zip	
<input type="checkbox"/>	14	Jeremy White	3954 Brentwood Dr	Seattle	WA	99037	  
<input type="checkbox"/>	15	Omar Badshah	6801 Regina Cir	Madison	WI	53209	  
<input type="checkbox"/>	16	Preston Harrison	104 Main St	Denver	CO	81712	  
<input type="checkbox"/>	17	Manuel Rodriguez	99543 Westin Blvd	Provo	UT	85478	  
<input type="checkbox"/>	18	David Jones	5488 W 34th St	Boston	MA	2104	  
<input type="checkbox"/>	19	Nick Flanders	3486 Happy Trails Dr	Springfield	OH	45872	  
<input type="checkbox"/>	20	Paul Brown	3290 Pennsylvania Ave	Chicago	IL	61208	  
<input type="checkbox"/>	21	Sara Rehm	7746 Wysong Ave	Detroit	MI	48913	  
<input type="checkbox"/>	22	Haley Carter	2957 Princess Way	Portland	OR	97532	  
<input type="checkbox"/>	23	Julian Thomas	5564 Dandy Trail	Santa Fe	NM	81543	  

Sorting the column data

```
SELECT city FROM customers ORDER BY city
```

Sorting multiple column data... in this query it first sort the cities and then sort the names according to the cities ....

```
SELECT state,name FROM customers ORDER BY state,name
```

«<T>>	state	name
<input type="checkbox"/>	AK	Corey Smith
<input type="checkbox"/>	AK	Ruth Bolen
<input type="checkbox"/>	AL	Crystal Jarvis
<input type="checkbox"/>	AL	Perry Jordan
<input type="checkbox"/>	AL	Thomas Jackson
<input type="checkbox"/>	AR	Katherine Cain
<input type="checkbox"/>	AZ	Debra Talkington
<input type="checkbox"/>	AZ	Sherry Gibbons
<input type="checkbox"/>	CA	Desmond Rafferty
<input type="checkbox"/>	CA	Donna Bradley
<input type="checkbox"/>	CA	Tack Nicholson

Reverse sorting of the values in the column.....

```
SELECT state,zip FROM customers ORDER BY zip DESC
```

<input type="checkbox"/>	AK	99831
<input type="checkbox"/>	WA	99753
<input type="checkbox"/>	WA	99037
<input type="checkbox"/>	OR	97532
<input type="checkbox"/>	CA	97221
<input type="checkbox"/>	CA	93980
<input type="checkbox"/>	HI	93525
<input type="checkbox"/>	CA	92953

```
SELECT state,zip FROM customers ORDER BY zip ASC (ascending to descending... it is done default so no need....)
```

Display the details of the customer who has the largest ID number and need to display only one name...

```
SELECT name,id FROM customers ORDER BY id DESC LIMIT 1
```

«T»	name	id
<input type="checkbox"/>	<input type="text"/> Lucy Bronson	96

Filtering stuffs.... Displaying particular data from the column with a specific data

```
SELECT id,name FROM customers WHERE id=37
```

«T»	id	name
<input type="checkbox"/>	37	Terry Mitchell

Displaying the values expect 37.....

```
SELECT id,name FROM customers WHERE id!=37
```

Displaying the values less than 7

```
SELECT id,name FROM customers WHERE id < 7
```

```
SELECT id,name FROM customers WHERE id <= 7
```

Displaying the values more than 7

```
SELECT id,name FROM customers WHERE id > 7
```

```
SELECT id,name FROM customers WHERE id >= 7
```

Displaying the values between the certain range.....

```
SELECT id,name FROM customers WHERE id BETWEEN 2 AND 5
```

«T»	id	name
<input type="checkbox"/>	2	Noah Parker
<input type="checkbox"/>	3	Kelsey Burger
<input type="checkbox"/>	4	Corey Smith
<input type="checkbox"/>	5	Harry Potter

Displaying the names who resides in STATE.....

```
SELECT name,state FROM customers WHERE state = 'CA'
```

«T»	name	state
<input type="checkbox"/>	Kelsey Burger	CA
<input type="checkbox"/>	Jack Nicholson	CA
<input type="checkbox"/>	Desmond Rafferty	CA
<input type="checkbox"/>	Paula Barker	CA
<input type="checkbox"/>	Donna Bradley	CA
<input type="checkbox"/>	Patsy Cline	CA

## AND OR conditions:

Displaying the elements which both condition are to be true...

```
SELECT id,name,state FROM customers WHERE state='FL'
```

<<T>>		<b>id</b>	<b>name</b>	<b>state</b>
<input type="checkbox"/>		12	Penny Green	FL
<input type="checkbox"/>		39	James Hamilton	FL
<input type="checkbox"/>		49	Evan Bayh	FL
<input type="checkbox"/>		55	Michael Orlando	FL

SELECT id,name,state FROM customers WHERE state='FL' AND city='Miami'

<<T>>		<b>id</b>	<b>name</b>	<b>state</b>
<input type="checkbox"/>		49	Evan Bayh	FL

Displaying the elements which any one of the condition is to be true...

SELECT name,city,state FROM customers WHERE city='Miami' OR state='FL'

<<T>>		<b>name</b>	<b>city</b>	<b>state</b>
<input type="checkbox"/>		Penny Green	Orlando	FL
<input type="checkbox"/>		James Hamilton	Ft Lauderdale	FL
<input type="checkbox"/>		Evan Bayh	Miami	FL
<input type="checkbox"/>		Michael Orlando	Pensacola	FL
<input type="checkbox"/>		Carol Brown	Ft Myers	FL
<input type="checkbox"/>		Tanya Hicks	Orlando	FL

## IN ----- IN OUT option:

SELECT name,state FROM customers WHERE state='CA' OR state='FL' OR state='NY'

This IN option helps in eliminating the OR bunches .....

SELECT name,state FROM customers WHERE state IN ('CA','FL','NY')

This NOT IN option displays the data other than the entered states....

SELECT name,state FROM customers WHERE state NOT IN ('CA','FL','NY')%

## SEARCHING IN DATABASE USING WILDCARD FILTER (%) option:

SELECT name FROM items WHERE name LIKE 'new%'

This % acts as the word which starts with “new” displays it .....

<<T>>		<b>name</b>
<input type="checkbox"/>		New gym socks
<input type="checkbox"/>		New ipad stolen from best buy
<input type="checkbox"/>		new curtain for bedroom
<input type="checkbox"/>		newspaper

SELECT name FROM items WHERE name LIKE '%computer%'

This displays the before and back words stacked to it....

<<T>>		<b>name</b>
<input type="checkbox"/>		Brand New iMac Computer
<input type="checkbox"/>		awesome alien computer game
<input type="checkbox"/>		supercomputer
<input type="checkbox"/>		computer

`SELECT city FROM customers WHERE city LIKE 'h%d'`

This displays the word which starts with “h” and ends with “d”

«T»			city
<input type="checkbox"/>			Hollywood
<input type="checkbox"/>			Highland

`SELECT name FROM items WHERE name LIKE '_ boxes of frogs'`

This command contains “\_” which displays only one character ....

«T»			name
<input type="checkbox"/>			3 boxes of frogs
<input type="checkbox"/>			7 boxes of frogs

`SELECT name FROM items WHERE name LIKE '% boxes of frogs'`

In table we have 48 boxes of frogs .... In order to display that too we need to use “%” wildcard

«T»			name
<input type="checkbox"/>			3 boxes of frogs
<input type="checkbox"/>			48 boxes of frogs
<input type="checkbox"/>			7 boxes of frogs

## SEARCHING IN DATABASE USING WILDCARD FILTER (%) REGULAR EXPRESSIONS:

`SELECT name FROM items WHERE name REGEXP 'new'`

This command displays everything which has a word “new”

«T»			name
<input type="checkbox"/>			Brand New iMac Computer
<input type="checkbox"/>			New gym socks
<input type="checkbox"/>			New ipad stolen from best buy
<input type="checkbox"/>			new curtain for bedroom
<input type="checkbox"/>			newspaper

`SELECT name FROM items WHERE name REGEXP '.boxes of frogs'`

This command “.” Plays as a space before “boxes of frogs”

«T»			name
<input type="checkbox"/>			3 boxes of frogs
<input type="checkbox"/>			48 boxes of frogs
<input type="checkbox"/>			7 boxes of frogs

`SELECT name FROM items WHERE name REGEXP 'gold | car'`

This command works a OR statement and displays the list which has either gold or car...

«T»			name
	✓	✗	traditional carpet
	✓	✗	gold necklace
	✓	✗	used car
	✓	✗	gold earring

SELECT name FROM items WHERE name REGEXP '[12345] boxes of frogs'

This command displays the set of numbers of “boxes of frogs” you may also use

SELECT name FROM items WHERE name REGEXP '[1-5] boxes of frogs'

«T»			name
	✓	✗	3 boxes of frogs

SELECT name FROM items WHERE name REGEXP '[^1-5] boxes of frogs'

This command displays the numbers other than the numbers in the set ....

«T»			name
	✓	✗	48 boxes of frogs
	✓	✗	7 boxes of frogs

## CREATING CUSTOM COLUMNS:

SELECT CONCAT(city,',',state) AS newAddress FROM customers

This command concate two columns into one column and “AS” displays a name to it ....

«T»	
newAddress	
Adams,NY	
Raleigh,NC	
Oakland,CA	
Simmersville,AK	
Newark,NJ	
Gary,IN	
Augusta,GA	

SELECT name,cost,cost+1 AS salePrice FROM items

This command adds a new column with addition of price in it ... you can use “+ - \* /” etc..

name	cost	salePrice
Brand New iMac Computer	149.99	150.990005493164
used diaper from my sister	2.04	3.03999996185303
Fresh apple pie	14.99	15.9899997711182
New gym socks	2.34	3.33999991416931
Weedwacker only slightly used	4.56	5.55999994277954
New ipad stolen from best buy	399	400
Book about having babies	21.34	22.3400001525879
Woman Jeans	49.5	50.5
traditional carpet	25.45	26.4500007629395
3 boxes of frogs	30.49	31.4899997711182

## FUNCTIONS:

SELECT name,UPPER(name) FROM customers

This command change the characters to uppercase....

«T»	
	UPPER(name)
Bucky Roberts	BUCKY ROBERTS
Noah Parker	NOAH PARKER
Kelsey Burger	KELSEY BURGER
Corey Smith	COREY SMITH
Harry Potter	HARRY POTTER

SELECT name,LOWER(name) FROM customers

This command change the characters to lowercase...

«T»	
	LOWER(name)
Bucky Roberts	bucky roberts
Noah Parker	noah parker
Kelsey Burger	kelsey burger

SELECT cost,SQRT(cost) FROM items

This command calculate the multiple operation of using SQRT

«T»	
	SQRT(cost)
149.99	12.247040683086
2.04	1.42828567235446
14.99	3.87169210696282

SELECT AVG(cost) FROM items

This command calculate the average

«T»	
	AVG(cost)
	463.937100627422

SELECT SUM(bids) FROM items

This command calculate the total of the numbers ...

«T»	
	SUM(bids)
	10939

## AGGREGATE FUNCTIONS:

SELECT COUNT(name) AS totalItems FROM items WHERE seller\_id=12

This command calculates the number of items (listings) selling by the customer...

totalItems
5

SELECT COUNT(\*) AS totalItems,

MAX(cost) AS maximum,

MIN(cost) AS minimum

FROM items WHERE seller\_id=12

This command calculates the min and max of the items sold by the customer

«T»

totalItems	maximum	minimum
5	5700.5	85.1999969482422

## GROUP BY -- HAVING:

SELECT seller\_id, COUNT(\*) AS countTable FROM items

«T»

seller_id	countTable
32	100

SELECT seller\_id, COUNT(\*) AS countTable FROM items WHERE seller\_id=4

«T»

seller_id	countTable
4	1

SELECT seller\_id, COUNT(\*) AS countTable FROM items WHERE seller\_id=3

SELECT seller\_id, COUNT(\*) AS countTable FROM items WHERE seller\_id=5

In order to avoid this bunch of codes ... we use single command which number of items sold by each customer

....

SELECT seller\_id, COUNT(\*) AS countTable FROM items GROUP BY seller\_id

«T»

seller_id	countTable
1	2
2	2
3	1
4	1
6	3

SELECT seller\_id, COUNT(\*) AS countTable FROM items GROUP BY seller\_id

HAVING count(\*)>3

This command lists the count table which the seller sells more than 3 items ....

«T»

seller_id	countTable
12	5

«T»

<u>seller_id</u>	<u>countTable</u>
14	4
15	4
18	4

SELECT seller\_id, COUNT(\*) AS countTable FROM items GROUP BY seller\_id

HAVING count(\*)>3 ORDER BY countable

This command sorts the countTable....

«T»

<u>seller_id</u>	<u>countTable</u>
18	4
15	4
14	4
12	5

## SUB QUERIES:

SELECT name,cost FROM items WHERE cost>(

SELECT AVG(cost) FROM items

) ORDER BY cost

This command displays the cost values more than the average (sub query)

«T»	<u>name</u>	<u>cost</u>
<input type="checkbox"/>	iphone	547
<input type="checkbox"/>	camera	550.7
<input type="checkbox"/>	electric oven	645
<input type="checkbox"/>	refrigerator	657.49

SELECT seller\_id FROM items WHERE name LIKE '%boxes of frogs'

«T»	<u>seller_id</u>
<input type="checkbox"/>	68
<input type="checkbox"/>	6
<input type="checkbox"/>	18

```
SELECT name,MIN(cost) FROM items WHERE name LIKE '%boxes of frogs'
```

```
AND seller_id IN(
```

```
 SELECT seller_id FROM items WHERE name LIKE '%boxes of frogs'
```

```
)
```

This command displays the cheapest cost of the item from the seller .....

«T»	
name	MIN(cost)
3 boxes of frogs	30.4899997711182

### Joining tables:

This command joins the two tables with some relations. For e.g.: customer id is related to the seller id. In this bucky Roberts sells two items.....

```
select customers.id,customers.name,items.name,items.cost
```

```
from customers,items
```

```
where customers.id = seller_id
```

```
ORDER by customers.id
```

id	name	name	cost
1	Bucky Roberts	used diaper from my sister	2.04
1	Bucky Roberts	bucket	2.5
2	Noah Parker	babyfoot	376.7
2	Noah Parker	baby seat	145.78
3	Kelsey Burger	lipstick	24.75
4	Corey Smith	baby soap	12.7
6	Henry Jackson	48 boxes of frogs	74.29
6	Henry Jackson	microwave	150.29
6	Henry Jackson	shampoing	12.8
7	Cynthia Alvarez	blue dress size 40	88.9
7	Cynthia Alvarez	scarf	11.9

### Nickname for tables:

```
Select i.seller_id, i.name, c.state, c.city
```

```
From customers as c, items as i
```

This commands helps in generating nicknames or assigning names for tables.

### OUTER joins:

This command generates all the users and their items still who doesn't list items in the database.

```
select customers.name, items.name from customers left outer join
```

```
items on customers.id = seller_id
```

RIGHT outer join displays the items still the seller is exited or banned from the website.

```
select customers.name, items.name from customers right outer join
```

```
items on customers.id = seller_id
```

name	name
Bucky Roberts	used diaper from my sister
Bucky Roberts	bucket
Noah Parker	baby seat
Noah Parker	babyfoot
Kelsey Burger	lipstick
Corey Smith	baby soap
Harry Potter	NULL
Henry Jackson	48 boxes of frogs
Henry Jackson	shampooing
Henry Jackson	microwave
Cynthia Alvarez	pan

LEFT

NULL	cushion
Lani Kulana	refrigerator
Sherry Gibbons	gold necklace
Cynthia Alvarez	pan

RIGHT

## UNION

This command displays the union of two queries into single column.

```
select name, cost, bids from items where cost>1000
```

```
UNION
```

```
select name, cost, bids from items where bids>190
```

name	cost	bids
conveyor belt	1120.75	4
used car	5700.5	135
piano	1800.4	147
machintosh	3845	107

```
select name, cost, bids from items where bids>190
```

```
UNION ALL
```

```
select name, cost, bids from items where cost>1000
```

This command displays the duplicates items too.

## Full Text Searching:

First you have write this query....

```
alter table items add fulltext(name)
```

Your SQL query has been executed successfully

```
select name,cost from items where match(name) against ('baby')
```

«T»			name	cost
			baby coat	89.99
			baby seat	145.78
			baby soap	12.7
			baby bottle	27.98

```
select name, cost from items where match(name) against('+baby -coat' in boolean mode)
```

This query searches for baby items and eliminates the coat word with the minus sign.....

«T»			name	cost
			baby seat	145.78
			baby soap	12.7
			baby bottle	27.98

## INSERT

This command helps in inserting a row or a column.

```
insert into items(id,name,cost,seller_id,bids) values('101','fish nuggets','10.20','1','10')
```

Inserted rows: 1

```
SELECT * FROM items
```

«T»			id	name	cost	seller_id	bids
			101	fish nuggets	10.2	1	10
			100	magazine	3.5	8	152
			99	dvd	126.84	14	113

## INSERT MULTIPLE ROWS:

This command helps in inserting multiple rows.

```
insert into items(id,name,cost,seller_id,bids) values
('102','beef','7.02','2','84'),
('103','shrimp','20.02','3','30'),
('104','pasta','2.02','4','20')
```

Inserted rows: 3

This command helps in inserting multiple rows from another table.

```
insert into items(id,name,cost,seller_id,bids)
```

```
Select id,name,cost,seller_id,bids from faketable
```

## UPDATE

This command updates the items in the row.

```
update items set name='beef pasta' where id=102
```

Affected rows: 1

«T»	id	name	cost	seller_id	bids
<input type="checkbox"/>	104	pasta	2.02	4	20
<input type="checkbox"/>	103	shrimp	20.02	3	30
<input type="checkbox"/>	102	beef pasta	7.02	2	84

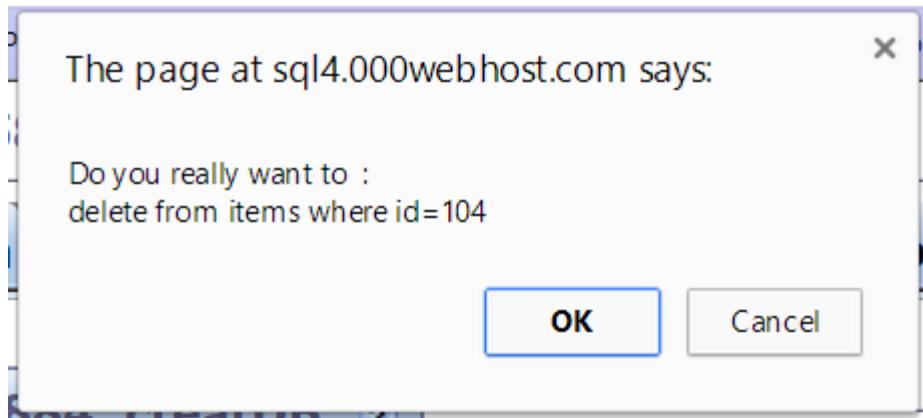
```
update items set name='beef pasta' bids = 100 where id=102
```

This command updates the items in the multiple columns.

## DELETE

This command deletes the row from the table.

```
delete from items where id=104
```



Deleted rows: 1

## CREATE A NEW TABLE

This command helps in creating a new table.

```
create table newtable(
 id int,
 username varchar(30),
 password varchar(20),
 primary key(id)
)
```

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	<b>id</b>	int(11)			No	0		
<input type="checkbox"/>	<b>username</b>	varchar(30)	latin1_general_ci		Yes	NULL		
<input type="checkbox"/>	<b>password</b>	varchar(20)	latin1_general_ci		Yes	NULL		

## NOT NULL AUTO\_INCREMENT

This command which has NOT NULL this will not allow to leave the username and password blank.....

This command which has AUTO\_INCREMENT this will increment automatically the id.....

```
create table freshtable(
 id int NOT NULL AUTO_INCREMENT,
 username varchar(20) NOT NULL,
 password varchar(30) NOT NULL,
 primary key(id)
)
```

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	<b>id</b>	int(11)			No		auto_increment	
<input type="checkbox"/>	<b>username</b>	varchar(20)	latin1_general_ci		No			
<input type="checkbox"/>	<b>password</b>	varchar(30)	latin1_general_ci		No			

«<T>>	<b>id</b>	<b>username</b>	<b>password</b>
	1	yunus	irshad
	2	zakira	banu

## ADDING COLUMN:

This command helps in adding a new column in the table.

```
alter table newtable add newcolumn varchar(30)
```

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	<b>id</b>	int(11)			No	0		
<input type="checkbox"/>	<b>username</b>	varchar(30)	latin1_general_ci		Yes	NULL		
<input type="checkbox"/>	<b>password</b>	varchar(20)	latin1_general_ci		Yes	NULL		
<input type="checkbox"/>	<b>newcolumn</b>	varchar(30)	latin1_general_ci		Yes	NULL		

## ALTER

```
alter table newtable drop column newcolumn
```

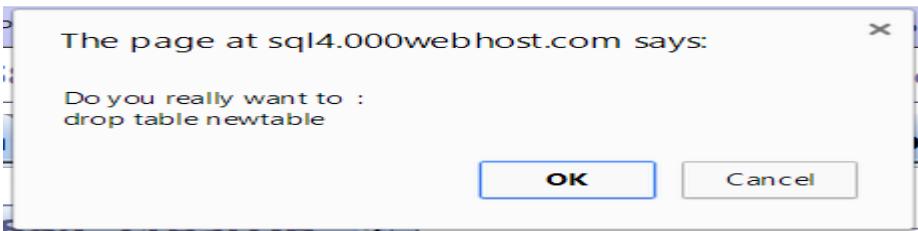
This command helps in dropping or deleting the new column from the table.



## DROP

drop table newtable

This command helps in dropping the table.



## RENAME

This command helps in renaming the table.

rename table customers to users

a5581884_create
(3)
■ freshtable
■ items
■ users

## VIEWS

This command is nothing but temporary view of the table of the command.

select concat(city, ', ', state) as newaddress from users

«»
newaddress
Adams, NY
Raleigh, NC
Oakland, CA
Simmersville, AK
Newark, NJ

create view mailing AS

a5581884_create
(4)
■ freshtable
■ items
■ mailing
■ users

```
select concat(city,',',state) as newaddress from users
```

[Google Checkout](#)[Pay Now](#)
[Home](#) [About Us](#) [Courses](#) [Services](#) [Clients](#) [Locations](#) [Contact](#) [Blogs](#)

by HtwokQA on December 1, 2012

[Leave a Comment](#)

## QA Blog

### Tips for QA Analyst Interview Questions

#### **Q. 101: What is the difference a Software Tester & Testing Analyst?**

Testing analysts are more commonly involved with tasks at a higher level of abstraction, such as test process design, test planning, and test case design.

Whereas Software Testers may be involved with test case design and test procedure construction, and interaction with the actual software systems.

&lt;&lt;&lt;&lt; ===== &gt;&gt;&gt;&gt;

#### **Q. 102: What are Software Testing Specialities?**

Testing specialties include test automation, load testing, usability testing, testing methodology, software inspections, industry or application expertise, test metrics, test management, white box testing & security testing etc.

&lt;&lt;&lt;&lt; ===== &gt;&gt;&gt;&gt;

#### **Q. 103: What can be the various Job Levels in the Software Testing Domain in a Company?**

Various job levels within the testing domain can include the tester, test analyst, test manager or test specialist, test consultant or Test executive.

&lt;&lt;&lt;&lt; ===== &gt;&gt;&gt;&gt;

#### **Q. 104: What is a Test Suite?**

Set of collection of test cases is called a test suite.

It contains more detailed instructions or goals for each collection of test cases. It contains a section where the tester identifies the system configuration used during testing. It may also contain prerequisite states or steps, and descriptions of the tests as well.

&lt;&lt;&lt;&lt; ===== &gt;&gt;&gt;&gt;

#### **Q. 105: What is a scenario test?**

This is a test based on a hypothetical story used to help a person think through a complex problem or system.

Generally scenario test have following five key characteristics.

- 1) A story

- 2) Which is motivating

#### Training Courses

- [QA Training Course](#)
- [BA Training Course](#)
- [Adv.QTP Training](#)
- [JAVA/J2EE Training](#)
- [.Net Training](#)
- [Informatica Training](#)

#### Latest Topics

- [Quality Assurance \(QA\) Online Training With Placement Assistance](#)
- [List of Quality Assurance Interview Questions 2013](#)
- [Tips for QA Analyst Interview Questions](#)
- [Frequently Asked Questions In Manual Testing](#)
- [Latest Quality Assurance Interview Questions With Solutions](#)

- 3) Which is credible
- 4) Which is complex
- 5) Which is easy to evaluate.

Scenario tests are different from test cases in a way that test cases cover single steps whereas scenarios cover a number of steps. Test suites and scenarios can be used together for a complete system test.

<<<< ===== >>>>

#### **Q. 106: What is Defect Tracking?**

In engineering practice, defect tracking is the process of finding defects in a product by the process of inspection, testing, or recording feedback from customers, and tracking them till their closure.

In software engineering, defect tracking is of significant importance, since complex software systems have thousands of defects due to which their management, evaluation and prioritizing is a difficult task. Hence defect tracking systems in software engineering are computer database systems which store defects and help people to manage them.

<<<< ===== >>>>

#### **Q. 107: What is Formal Verification in context with Software & Hardware systems?**

Formal verification is the process of proving or disproving the correctness of a system with respect to a certain formal specification or property, with the help of formal methods. Generally the formal verification is carried out algorithmically.

Approaches to implement formal verification are :

- 1) State space enumeration
- 2) Symbolic state space enumeration
- 3) Abstract interpretation
- 4) Abstraction refinement
- 5) Process-algebraic methods
- 6) Reasoning with the help of automatic theorem provers like HOL or Isabelle.

<<<< ===== >>>>

#### **Q. 108: What is the concept of Fuzz Testing?**

Fuzz testing is a software testing technique involving attachment of the inputs of a program to a source of random data. Main advantage of fuzz testing is that the test design is extremely simple, and remains free of preconceptions about system behavior.

Fuzz testing is generally used in large software development projects which use black box testing. Fuzz testing provides a high benefit to cost ratio.

Fuzz testing technique is also used for the measurement of quality of large software systems. The advantage is that the cost of generating tests is relatively low.

Fuzz testing helps to enhance the software security and software safety because it often finds odd oversights and defects which normal human testers would fail to find, and even the most careful human test designers would fail to create tests for.

Fuzz testing is not a substitute for exhaustive testing or formal methods; it can only provide a random sample of the system's behavior. Passing a fuzz test may only indicate that a particular software is capable of handling exceptions without crashing and it may not indicate its correct behavior.

<<<< ===== >>>>

#### **Q. 109: What are the different forms of fuzz testing?**

- 1) Valid fuzz Testing to assure that the random input is reasonable, or conforms to actual production data.
- 2) Simple fuzz Testing usually uses a pseudo random number generator to provide an input.
- 3) A combined approach uses valid test data with some proportion of totally random input injected.

By using all the above techniques in combination, fuzz-generated randomness can test the un-designed behavior surrounding a wider range of designed system states.

<<<< ===== >>>>

#### **Q. 110: What is a Web Application & How does it look like?**

A web application is an internet based application, consisting of a set of many scripts, which are normally stored on some Web server and are made to interact with some databases or any other similar sources of the dynamic content.

Web applications provide an interactive Form to the user, wherein feeds inputs according to the fields provided in the form; then he clicks on a button like "Submit" or "OK" to store his inputs on the database & perform a set of calculations & present back the desired information.

Web Applications are becoming popular since these are a via media for exchange of information between various service providers and respective customers across the internet. These web applications are by & large not dependent on any platform. Popular examples of Web applications are Google / Yahoo or similar search engines, Internet Banking websites of several Banks, E-mail facility providing sites like Gmail, Yahoo Mail, Rediff Mail etc., Sale & Purchase sites like E-Bay etc.

Request for FREE DEMO @ [QA Online Training](#).

---

 Filed Under: [Testing FAQ](#)  Tagged With: [faq in qa](#), [faq in qa 2013](#), [list of qa interview questions 2013](#), [qa interview questions and answers 2013](#), [qa online training demo schedules](#), [topics covered in qa](#)

### **Speak Your Mind**

<input type="text"/>	Name *
<input type="text"/>	Email *
<input type="text"/>	Website

[Post Comment](#)

[Return to top of page](#)

Copyright © 2014 · Copyblogger Theme on Genesis Framework · WordPress · Log in

# XHTML -- CSS

XHTML is HTML written as XML.

- XHTML stands for EXtensible HyperText Markup Language
- XHTML is almost identical to HTML
- XHTML is stricter than HTML
- XHTML is HTML defined as an XML application
- XHTML is supported by all major browsers

## Why XHTML?

Many pages on the internet contain "bad" HTML.

The following HTML code works fine in most browsers (even if it does NOT follow the HTML rules):

```
<html>
<head>
<title>This is bad HTML</title>
<body>
<h1>Bad HTML
<p>This is a paragraph
</body>
```

XML is a markup language where documents must be marked up correctly (be "well-formed")

## The Most Important Differences from HTML:

<!DOCTYPE ....> Is Mandatory

An XHTML document must have an XHTML DOCTYPE declaration.

A complete list of all the [XHTML Doctypes](#) is found in our HTML Tags Reference.

The <html>, <head>, <title>, and <body> elements must also be present, and the xmlns attribute in <html> must specify the xml namespace for the document.

The example below shows an XHTML document with a minimum of required tags:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<title>Title of document</title>
</head>
```

```
<body>
.....
</body>

</html>
```

## XHTML Elements Must Be Properly Nested

In HTML, some elements can be improperly nested within each other, like this:

```
<i>This text is bold and italic</i>
```

In XHTML, all elements must be properly nested within each other, like this:

```
<i>This text is bold and italic</i>
```

## XHTML Elements Must Always Be Closed

This is wrong:

```
<p>This is a paragraph
<p>This is another paragraph
```

This is correct:

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

## Empty Elements Must Also Be Closed

This is wrong:

```
A break:

A horizontal rule: <hr>
An image:
```

This is correct:

```
A break:

A horizontal rule: <hr />
An image:
```

## XHTML Elements Must Be In Lower Case

This is wrong:

```
<BODY>
<P>This is a paragraph</P>
</BODY>
```

This is correct:

```
<body>
<p>This is a paragraph</p>
</body>
```

## XHTML Attribute Names Must Be In Lower Case

This is wrong:

```
<table WIDTH="100%">
```

This is correct:

```
<table width="100%">
```

## Attribute Values Must Be Quoted

This is wrong:

```
<table width=100%>
```

This is correct:

```
<table width="100%">
```

## Attribute Minimization Is Forbidden

This is wrong:

```
<input checked>
<input readonly>
<input disabled>
<option selected>
```

This is correct:

```
<input checked="checked">
<input readonly="readonly">
<input disabled="disabled">
<option selected="selected">
```

## How to Convert from HTML to XHTML

1. Add an XHTML <!DOCTYPE> to the first line of every page
2. Add an xmlns attribute to the html element of every page
3. Change all element names to lowercase
4. Close all empty elements
5. Change all attribute names to lowercase
6. Quote all attribute values

# CSS

## font-weight & font-style

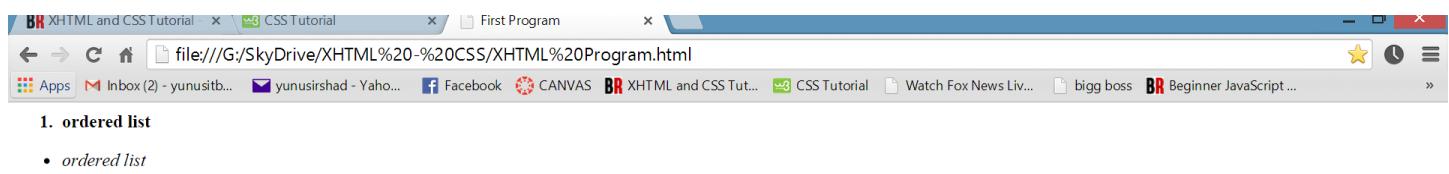
```
<style>
 ol {font-weight:bold} // it doesn't support italics
 ul {font-style:italic} // it doesn't support bold
</style>

</head>
<body>

 ordered list

 ordered list

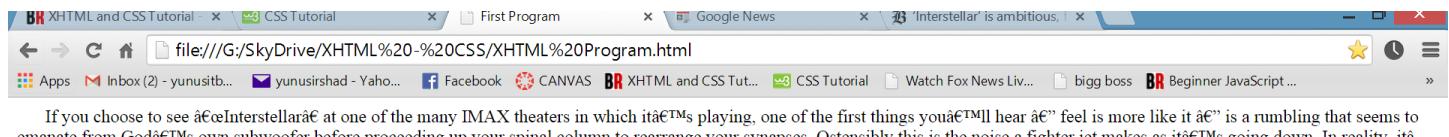
```



## Text-indent:

```
<style>
 p {text-indent: 25px} // space at the front of the paragraph
</style>

</head>
<body>
 <p> If you choose to see "Interstellar" at one of the many IMAX theaters in which it's playing, one of the first things you'll hear — feel is more like it — is a rumbling that seems to emanate from God's own subwoofer before proceeding up your spinal column to rearrange your synapses. Ostensibly this is the noise a fighter jet makes as it's going down. In reality, it's the sound of Christopher Nolan's ambition.
 </p>
```



## Background Images

```
<style>
```

```

body {background-image: url(logo.png); //background image with repeats
 background-repeat: no-repeat; //background image with no repeats
 background-repeat: repeat-x; //background image repeats in horizontal direction
 background-repeat: repeat-y; //background image repeats in vertical direction
 background-position: 50% 60px; //width and height of the image
 background-position: 50px 60px
}
</style>

```

## PADDING:

```

<style>
 h1 {background-color:yellow; padding-bottom:20px} //padding at the bottom
 h2 {background-color:yellow; padding-top:20px} //padding at the top
 h3 {background-color:yellow; padding-left:20px} //padding at the left
 h4 {background-color:yellow; padding-right:20px} //padding at the right
</style>
</head>
<body>
 <h1>This is my padding</h1>

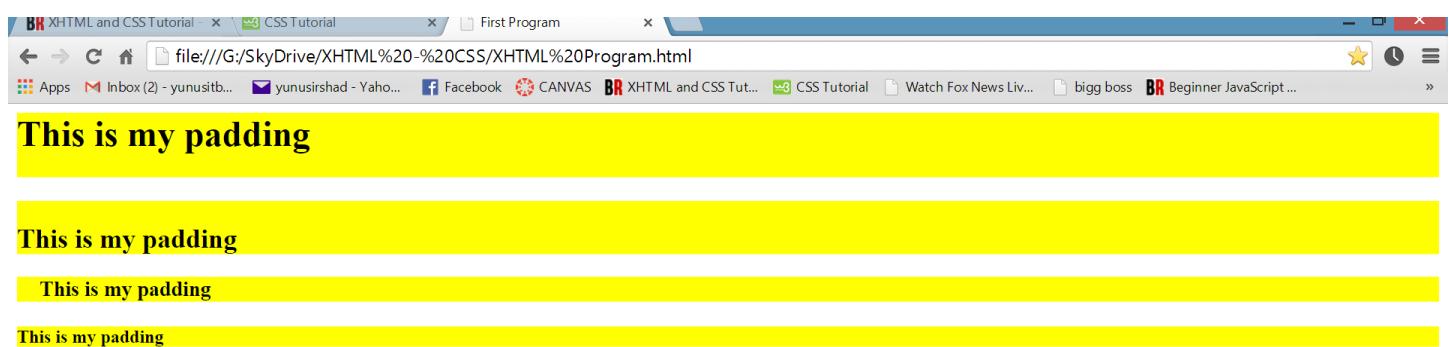
 <h2>This is my padding</h2>

 <h3>This is my padding</h3>

 <h4>This is my padding</h4>

</body>

```



## BORDERS:

```

<style>
 h1 {background-color:yellow; padding:20px;
 border-style:dashed; border-color:red; border-width:4px}
 h2 {background-color:yellow; padding:40px;
 border-bottom-color:dark red; // color of border
}
</style>

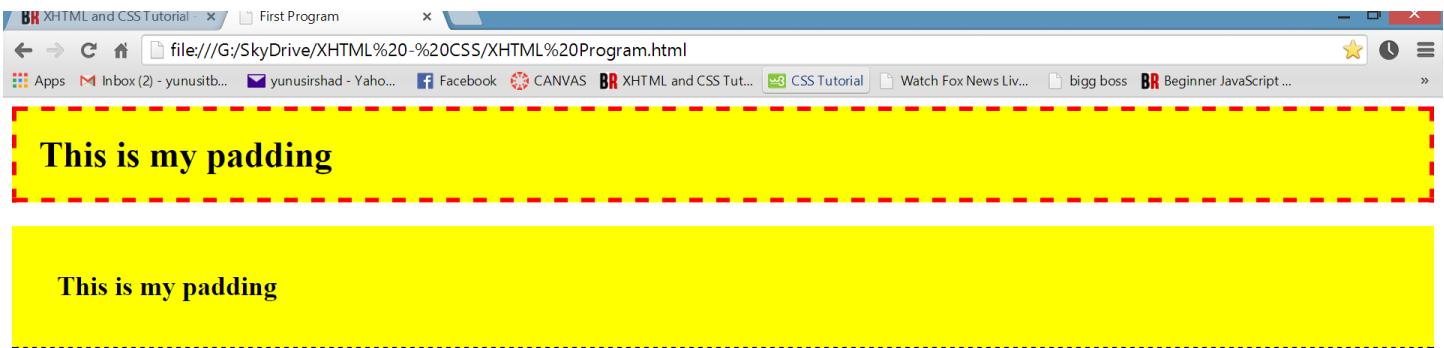
```

```

border-bottom-style:dotted; // style of border
border-bottom-width:4px // width of border
}

</style>

```



## MARGINS:

- Outside the border or padding.....

```

<style>
 h1,h2 {margin:40px}
</style>

```



**This is my padding**

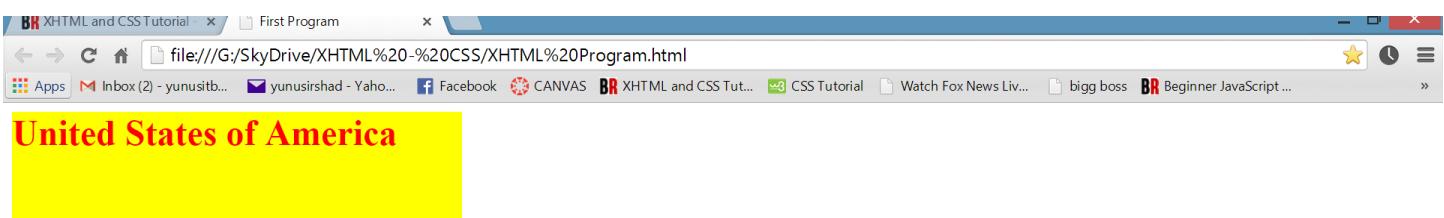
**This is my padding**

## WEIGHT & HEIGHT:

```

<style>
 h1{background-color:yellow; color:red; width:400px; height:100px}
</style>

```



## STYLING LINKS:

```
<style>
```

```

a:link{color:GREEN; text-decoration:none}
 // link will be visible in green color and text decoration none which means no underline.

a:visited {color:red}
 // link which is visited and is in red color

a:hover{background-color:yellow; color:red; text-decoration:underline}
 // the color changes when the mouse cursor goes on the link

a:active{background-color:orange} //if you hold the mouse cursor on the link it will go orange...

```

</style>

<a href="http://www.google.com">GOOGLE</a>

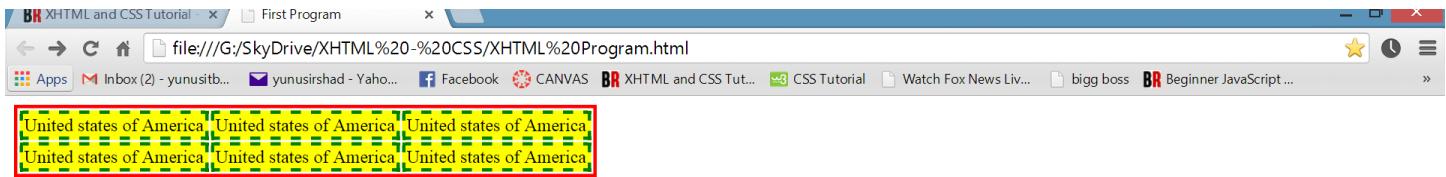
## STYLING TABLES:

```

<style>
 table{border:3px solid red} //table styling
 tr {background-color:yellow} //row styling
 td {border-style:dashed; border-color:green} //column styling

```

</style>



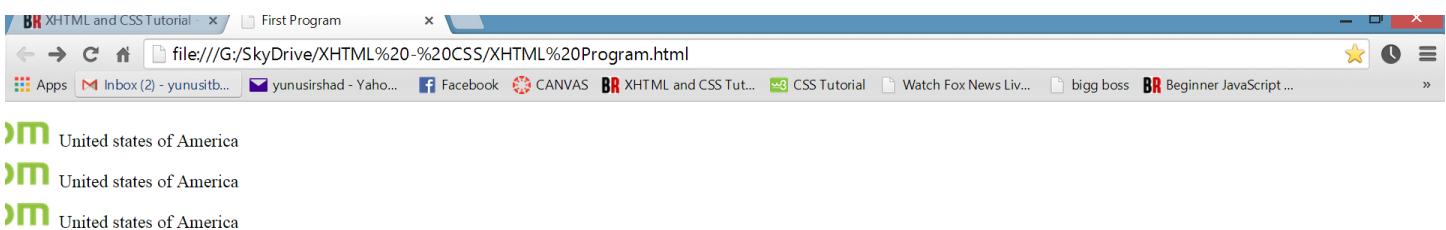
## STYLING UNORDERED LISTS:

```

<style>
 ul {list-style-type:none; // this displays no bullet points
 list-style-image:url(logo.png)} // this displays image instead of bullet points

```

</style>



## STYLING A ELEMENT INBETWEEN SENTENCE USING SPAN

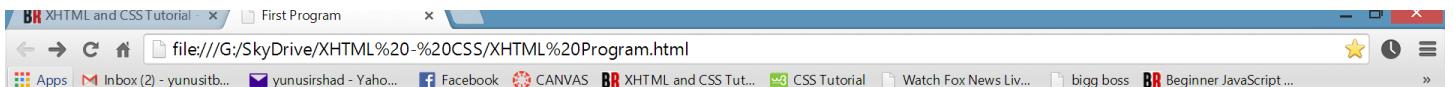
```

<style>
 span{color:red; font-weight:bold}

```

</style>

<p>United states of <span>America</span></p>



United states of America

## <div>

- Section of the website

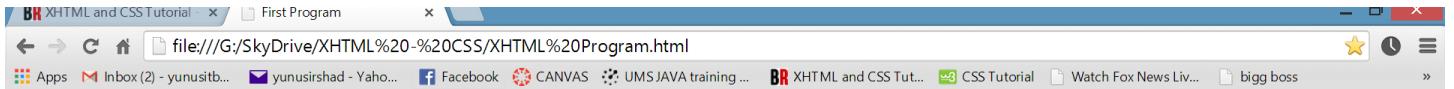
```
<style>
```

```
div{border:4px solid red; width:200px; position:absolute; top:100px; left:100px}
```

// without position type : you can't see the top and left positions....

```
</style>
```

```
<div>United states of America</div>
```



United states of America

## STYLING USING CLASS:

- Making two paragraphs in different colors.

```
<style>
```

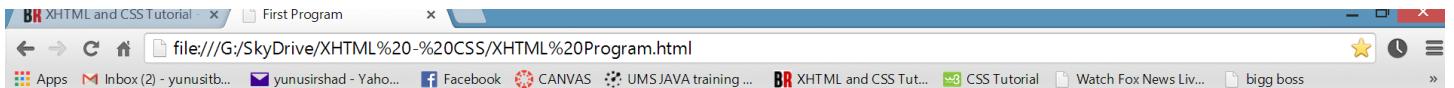
```
.redClass{color:red}
```

```
.greenClass{color:green}
```

```
</style>
```

```
<p class="redClass">RED: United states of America</p>
```

```
<p class="greenClass">GREEN: United states of America</p>
```



RED: United states of America

GREEN: United states of America

## STYLING USING ID:

- It is created to make it in different locations on the website.

```
<style>
```

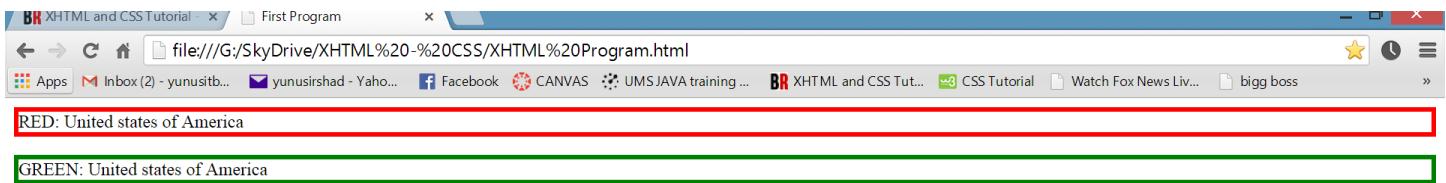
```
#redClass{border: 4px solid red}
```

```
#greenClass{border: 4px solid green}
```

```
</style>
```

```
<p id="redClass">RED: United states of America</p>
```

```
<p id="greenClass">GREEN: United states of America</p>
```

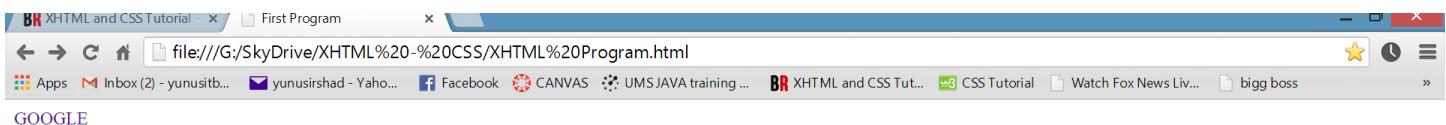


## STYLING USING CHILD:-

- Styling inside one another....

```
<style>
 p > a {font-size:100px;color:red} // children of the "a" that is "p" is styled
</style>

GOOGLE
<p>PARAGRAPH GOOGLE</p>
<h1>HEADER GOOGLE</h1>
```



# PARAGRAPH GOOGLE

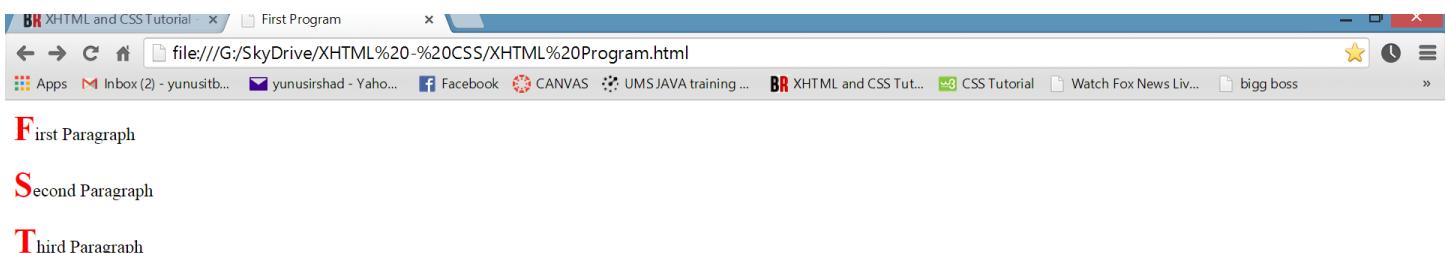
## HEADER GOOGLE

## STYLING USING PSEUDO ELEMENTS:-

- Styling the piece of an element. Such as capitalizing the beginning of the paragraph.

```
<style>
 p:first-letter{color:red; font-size:30px; font-weight:bold}
</style>

<p>First Paragraph</p>
<p>Second Paragraph</p>
<p>Third Paragraph</p>
```



## EXTERNAL STYLES:

- Styling one or more pages by creating styles.css

### XHTML PROGRAM.html

```
<link rel="stylesheet" href="Styles.css">
```

```
<h1>This is INDEX page</h1>
<p>This the paragraph</p>
Second page
```

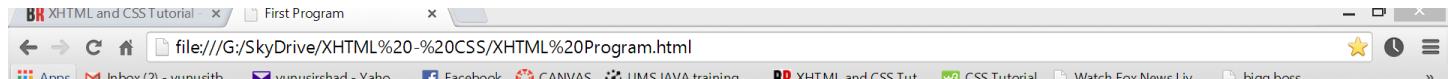
### XHTML PROGRAM2.html

```
<link rel="stylesheet" href="Styles.css">
```

```
<h1>This is SECOND page</h1>
<p>This the paragraph</p>
FIRST page
```

### Styles.css

```
h1 {color:red; font-family:Georgia}
p {color:blue; font-size:bold}
a {color:green; font-weight:20px}
```



## This is INDEX page

This the paragraph

[Second page](#)



## OVERRIDING:

- Styling one page from the bunch of 500 pages... CSS takes place the color which is finally declared.

```
<head>
```

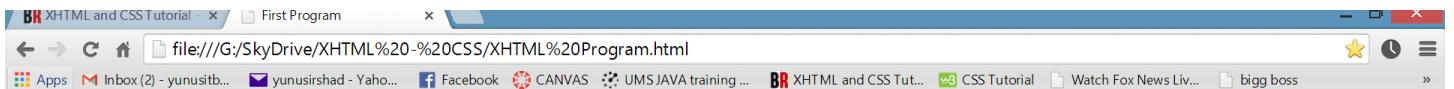
```
 <link rel="stylesheet" href="Styles.css">
```

```
 <style>
```

```
 h1 {color:Orange}
```

```
 </style>
```

```
</head>
```



## This is INDEX page

[This the paragraph](#)

[Second page](#)

## ABSOLUTE POSITIONING:

- The positioning takes place from the top left corner border.

<style>

```
#SECOND {border:4px solid red;
position:absolute;
height:150px;
width:200px;
top:30px;
left:40px}
```

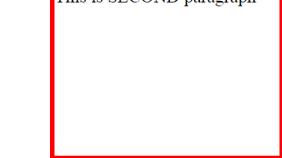
</style>

```
<p>This is FIRST paragraph</p>
<p id="SECOND">This is SECOND paragraph</p>
<p>This is THIRD paragraph</p>
```



This is FIRST paragraph

This is SECOND paragraph



## RELATIVE POSITIONING:

- The positioning takes place from the place where it actually declared instead of top left corner border

<style>

```
#SECOND {border:4px solid red;
position:absolute;
height:150px;
width:200px;
top:30px;
left:40px}
```

</style>

```
<p>This is FIRST paragraph</p>
<p id="SECOND">This is SECOND paragraph</p>
<p>This is THIRD paragraph</p>
```

This is FIRST paragraph

This is SECOND paragraph

This is THIRD paragraph

## FIXED POSITIONING:

- The positioning is fixed at the place where it actually declared using top and left pixels.

```
<style>
#SECOND {border:4px solid red; top:25px; left:45px; position:fixed}
</style>
<p>This is FIRST paragraph</p>
<p id="SECOND">This is SECOND paragraph</p>
<p>This is THIRD paragraph</p>
<p>This is THIRD paragraph</p>.....
```

This is FIRST paragraph

This is THIRD paragraph

This is THIRD paragraph

This is THIRD paragraph

This is THIRD paragraph

## MAX WIDTH-HEIGHT:

- You can set your images maximum width and height. Such as resizing images in ebay in same size

```
<style>
img {max-height:200px; max-width:200px}
</style>


```

- Same as you can set minimum width and height....

w3schools.com



