JAVA

TOPS Technologies

JAVA

- Module 1
  - Fundamental
- Module 2
  - Learning CoreJAVA
- Module 3
  - RDBMS & DatabaseProgramming using JDBC
- Module 4
  - Web Technologies in Java
- Module 5
  - Applicability to Industry
- Module 6
  - Hibernate
- Module 7
  - Spring Framework
- Module 8
  - Applicability of Industrial Project

Module – 1[Fundamental]

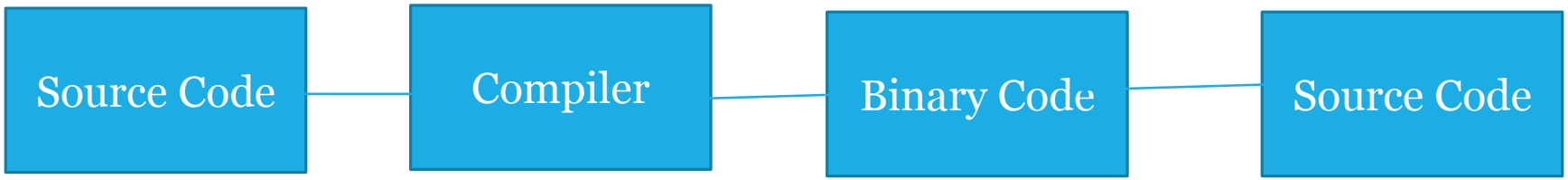
- **Introduction of Programming**
- **Conditional Statement**
- **Loops**
- **SDLC Process**
- **Project Analysis**
- **DFD**
- **Flow Chart**

TOPS Technologies

Basic Programming

- **Program**:
  - A computer program is a collection of instructions that performs a specific task when executed by a computer.
- A computer program is usually written in a programming language.
- Program Can be written in higher level programming languages that human can understand.
- Those programs are translated to computer understandable languages.
- Task will be done by special tool called compiler.

- **Compiler**:
  - Compiler will convert higher level language code to lower language code like binary code

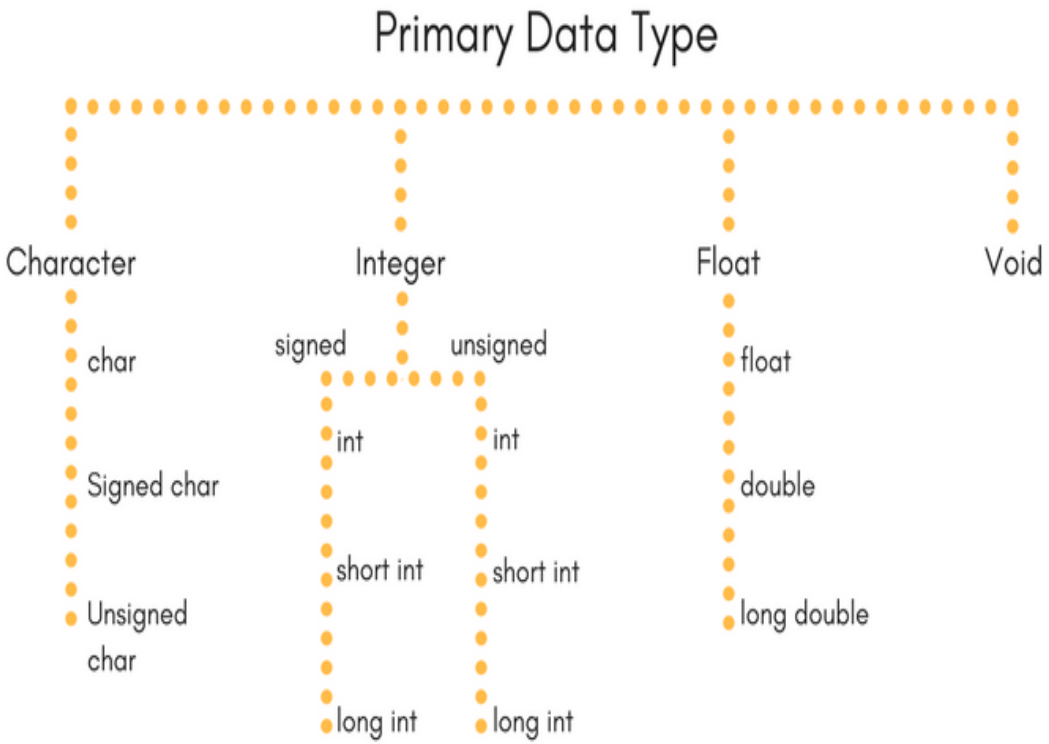


TOPS Technologies

- **Syntax**:
  - Rules & format that should be followed while writing program are called syntax.
- **Data Types**
  - Each variable in C has an associated data type.
  - Each data type requires different amounts of memory and has some specific operations which can be performed over it.
- **Identifiers**
  - Identifier refers to name given to entities such as variables, functions, structures etc.
  - Identifier must be unique. They are created to give unique name to a entity to identify it during the execution of the program.

- Rules for writing an identifier:
    1. A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.
    2. The first letter of an identifier should be either a letter or an underscore. However, it is discouraged to start an identifier name with an underscore.
    3. There is no rule on length of an identifier. However, the first 31 characters of identifiers are discriminated by the compiler.

- **Keywords**
    - Keywords are predefined, reserved words used in programming that have special meanings to the compiler.
    - Keywords are part of the syntax and they cannot be used as an identifier.



Primary Data Types

TOPS Technologies

Type	Size(bytes)	Range
int or signed int	2	-32,768 to 32767
unsigned int	2	0 to 65535
short int or signed short int	1	-128 to 127
unsigned short int	1	0 to 255
long int or signed long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295

Type	Size(bytes)	Range
Float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
long double	10	3.4E-4932 to 1.1E+4932

Type	Size(bytes)	Range
char or signed char	1	-128 to 127
unsigned char	1	0 to 255

Void type means no value. This is usually used to specify the type of functions which returns nothing.

Primary Data Types

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Keywords

TOPS Technologies

- **Operators**
  - C programming has various operators to perform tasks including arithmetic, conditional and bitwise operations.
  - **Operators in C programming**
    - Arithmetic Operators
    - Increment and Decrement Operators
    - Assignment Operators
    - Relational Operators
    - Logical Operators
    - Conditional Operators
    - Bitwise Operators
    - Special Operators

C Arithmetic Operators

Operator	Meaning of Operator
+	addition or unary plus
-	subtraction or unary minus
*	multiplication
/	division
%	remainder after division(modulo division)

C Assignment Operators

Operator	Example	Same as
=	a = b	a = b
+=	a += b	a = a+b
-=	a -= b	a = a-b
*=	a *= b	a = a*b
/=	a /= b	a = a/b
%=	a %= b	a = a%b

C Relational Operators

Operator	Meaning of Operator	Example
==	Equal to	5 == 3 returns 0
>	Greater than	5 > 3 returns 1
<	Less than	5 < 3 returns 0
!=	Not equal to	5 != 3 returns 1
>=	Greater than or equal to	5 >= 3 returns 1
<=	Less than or equal to	5 <= 3 return 0

C Logical Operators

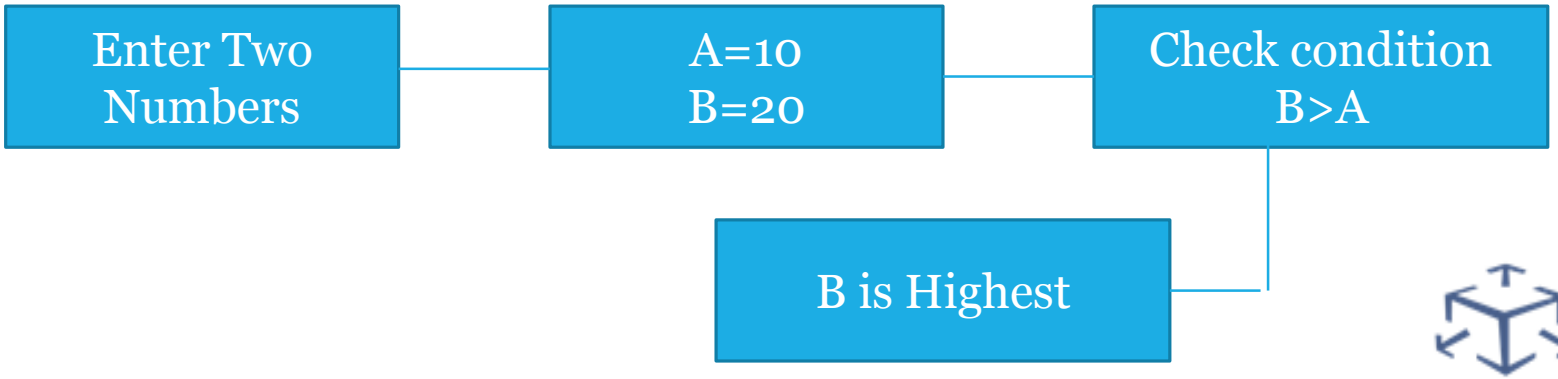
Operator	Meaning of Operator	Example
&&	Logial AND. True only if all operands are true	If c = 5 and d = 2 then, expression ((c == 5) && (d > 5)) equals to 0.
||	Logical OR. True only if either one operand is true	If c = 5 and d = 2 then, expression ((c == 5) || (d > 5)) equals to 1.
!	Logical NOT. True only if the operand is 0	If c = 5 then, expression ! (c == 5) equals to 0.

- **Conditional Statements**

- if a Condition is evaluates to true then the code block will be executed.
  - If Statement
    - Syntax:

```
if (Condition)
{
        Statements will execute on true condition
}
```

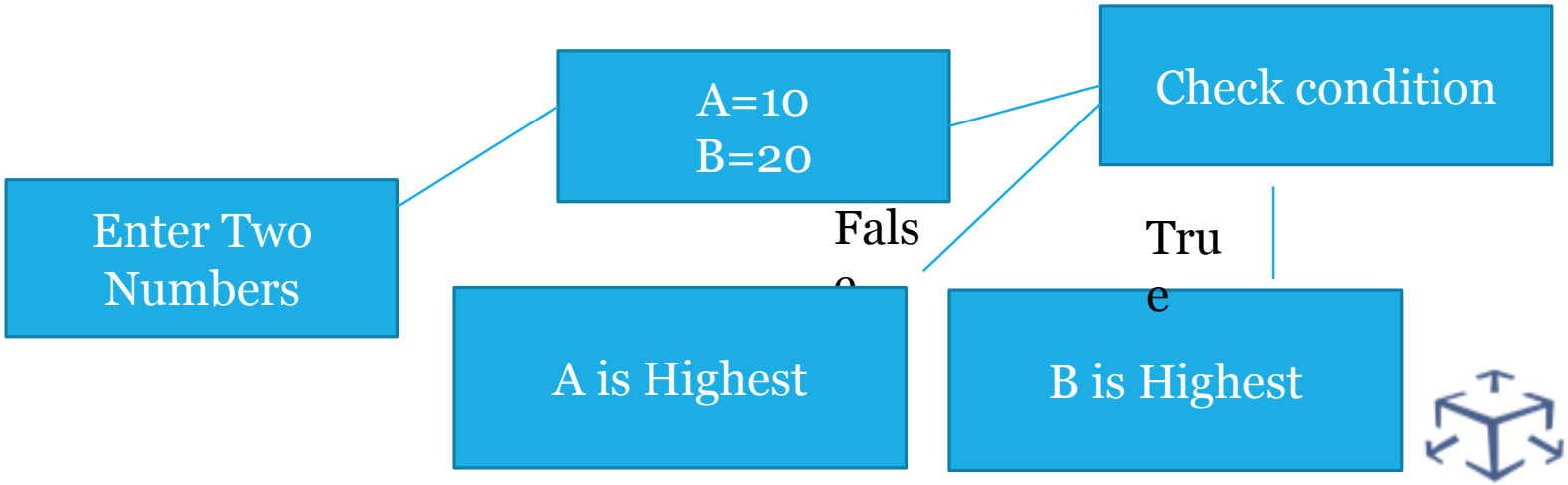
- **Check Highest Number in Given Two numbers.**



The **if/else statement** executes a block of code **if** a specified condition is true. **If** the condition is false, another block of code can be executed.

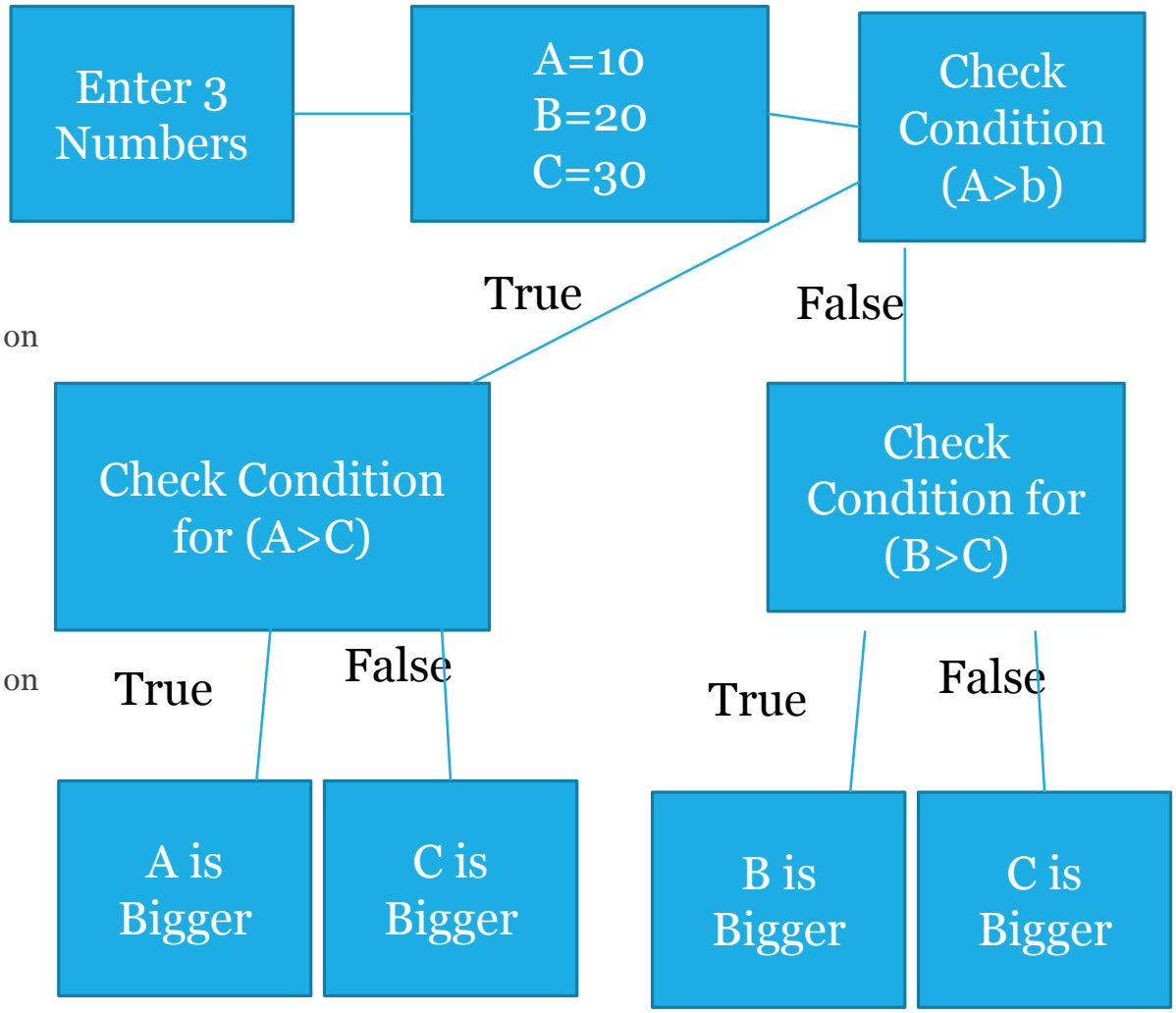
- If Else Statement
  - Syntax:

```
if (Condition)
{
        Statements will execute on true condition
}
else
{
        Statements will execute on false condition
}
```



- Conditional Statements
  - Nested IF
    - Syntax:

```
if (Condition-1)
{
    if (Condition-1)
    {
            Statements will execute on
        true condition-1
    }
}
else if(condition-2)
{
    if (Condition-1)
    {
            Statements will execute on
        true condition-1
    }
}
else
{  ...
}
```



- **Looping Statements**

a loop is a sequence of instruction s that is continually repeated until a certain condition is reached.

Loop Type & Description

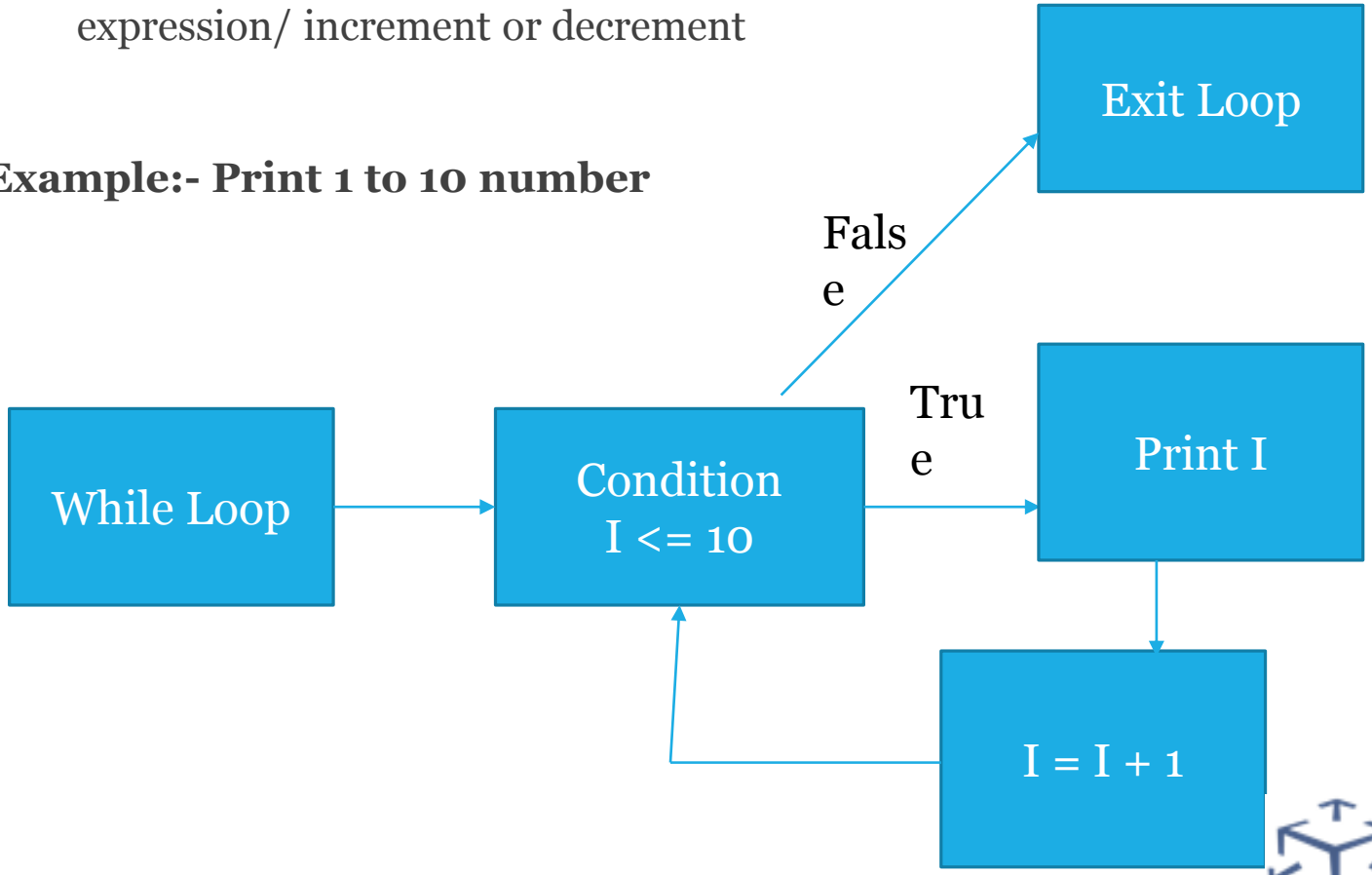
while loop	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
for loop	Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
do...while loop	It is more like a while statement, except that it tests the condition at the end of the loop body.

TOPS Technologies

- Looping Statements
  - While Loop
    - Syntax:

```
while( condition)
{
        Loop statements until the codition is true loop will execute
        expression/ increment or decrement
}
```

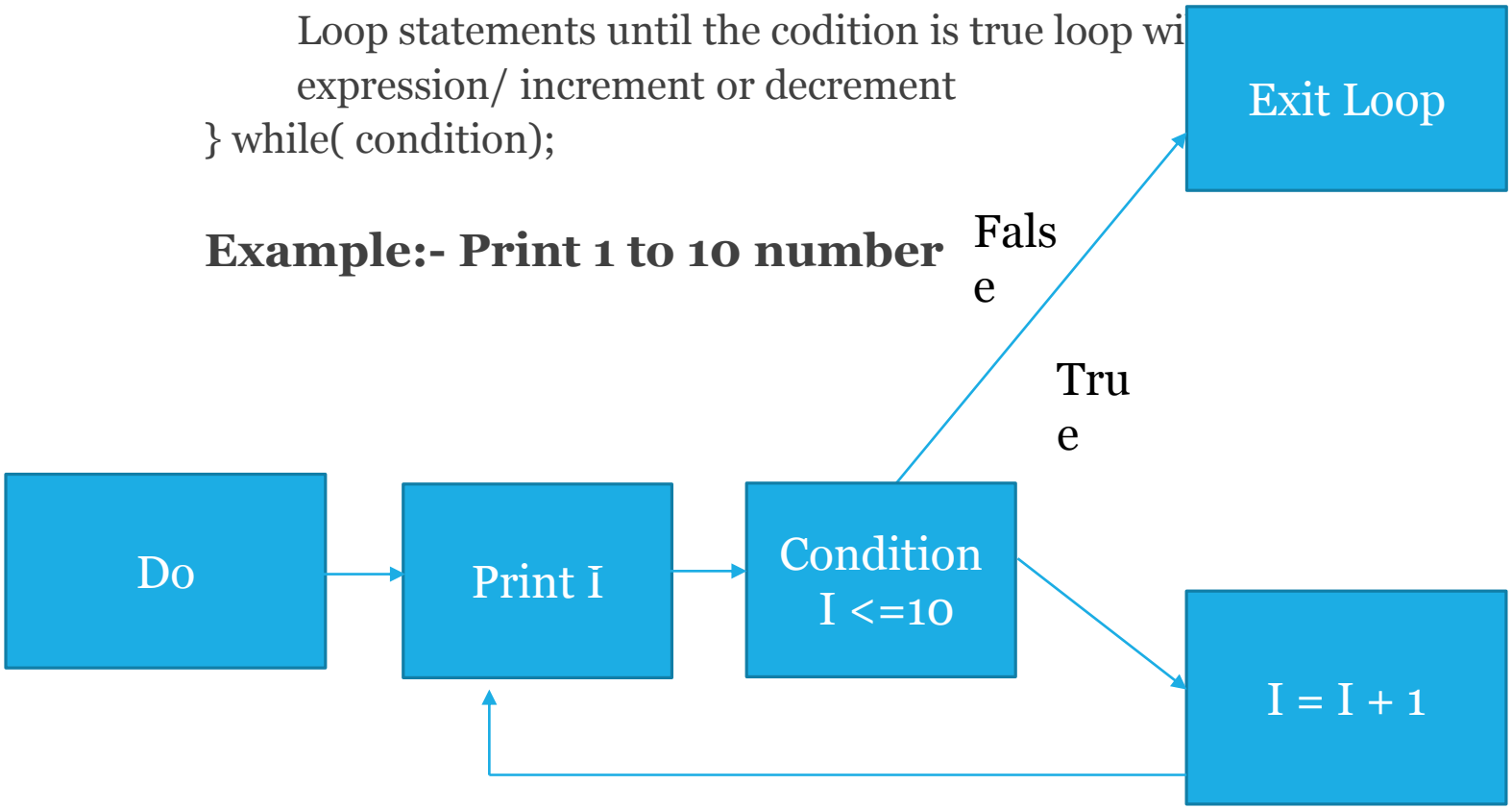
Example:- Print 1 to 10 number



- Do while loop print output at least 1 time when condition is false or True.
  - Do While Loop
    - Syntax:

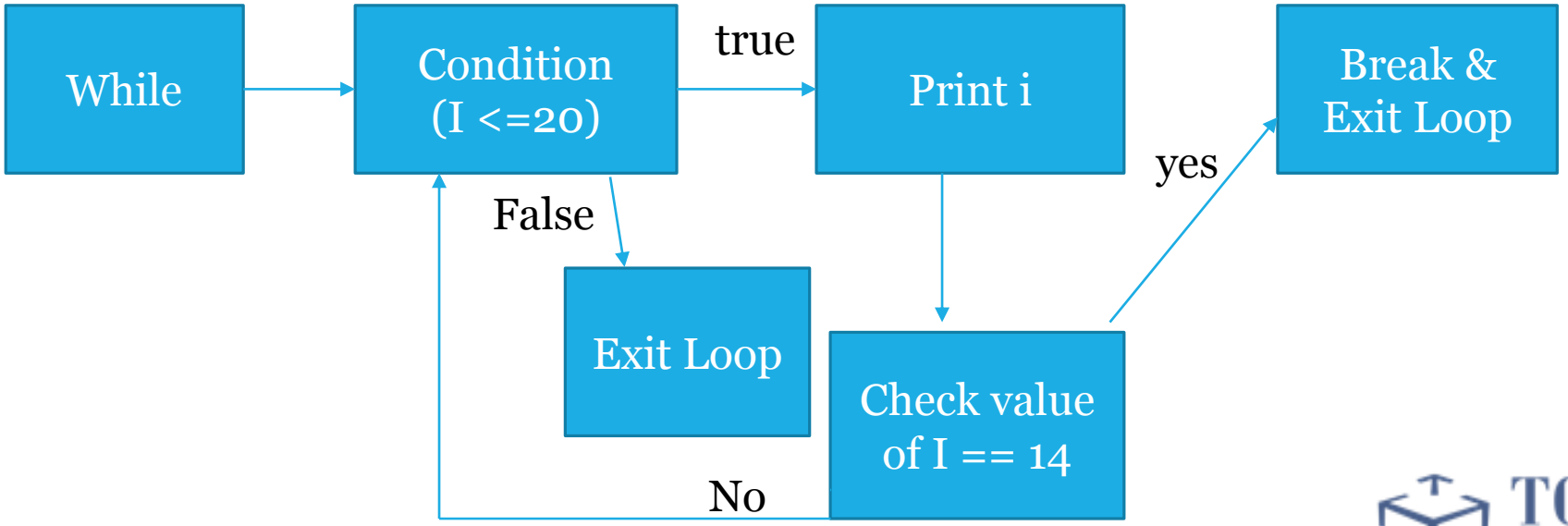
```
do
{
        Loop statements until the codition is true loop wi
        expression/ increment or decrement
} while( condition);
```

Example:- Print 1 to 10 number



- **Break:-**
- When a **break** statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.

- Run loop until value of I is 20, Break statement when value of I is 14.
- Value of I=1;

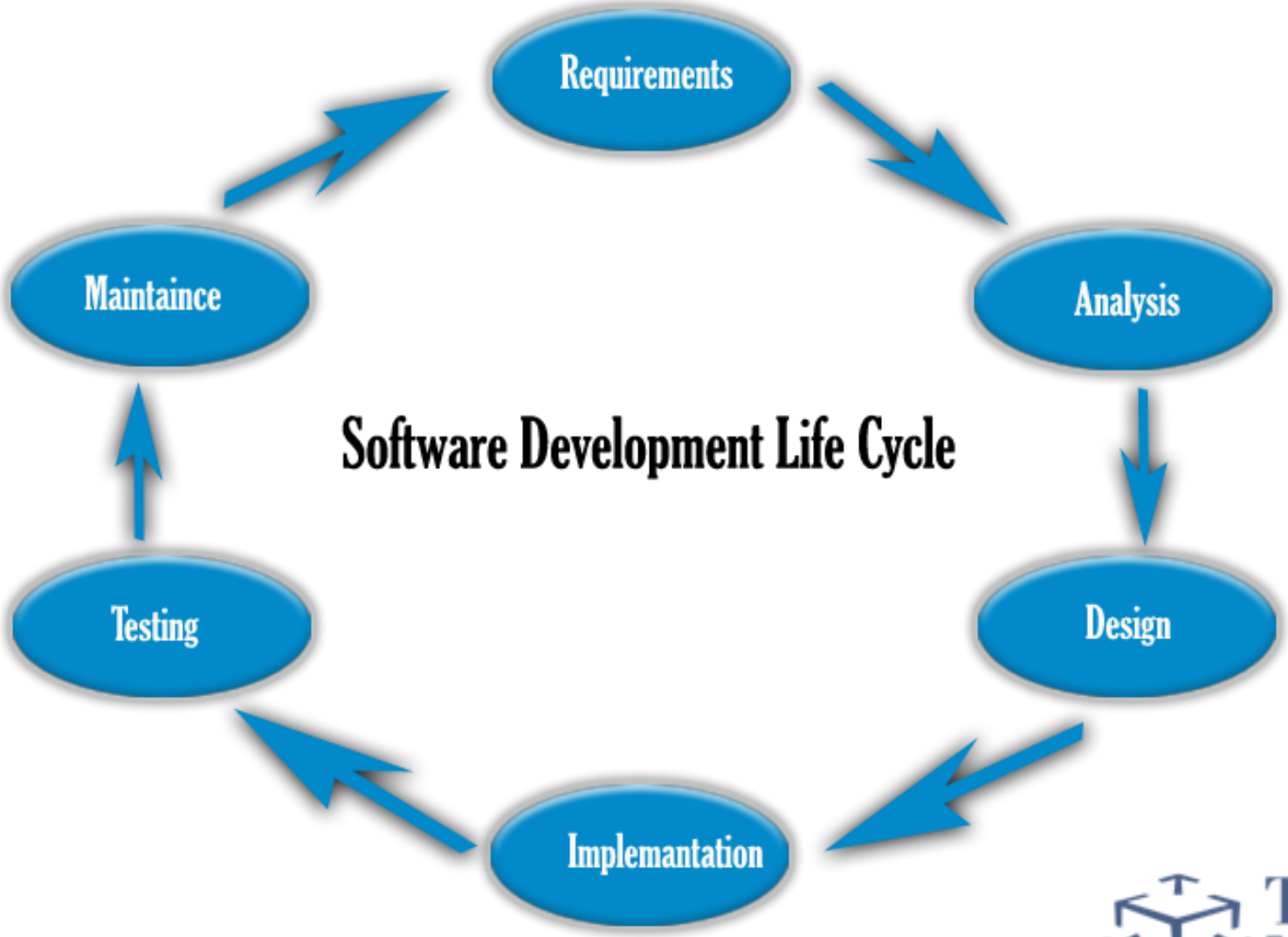


SDLC

- For project development rules & regulation need to be followed for best quality output at defined time limit
- Rules are Software Development Life Cycle – SDLC
- It’s part of software engineering
- Six rules to be followed…

TOPS Technologies

- Requirement Gathering
- Analysis & SRS
- Designing
- Implementation (Coding)
- Testing
- Maintenance



TOPS Technologies

DFD

- DFD – Data Flow Diagrams
- Graphical representation of flow of data inside application can also be used for visualization and data processing
- DFD elements are..
  - External Entity
  - Process
  - Data Flow
  - Data Store

TOPS Technologies

1) **External entity**

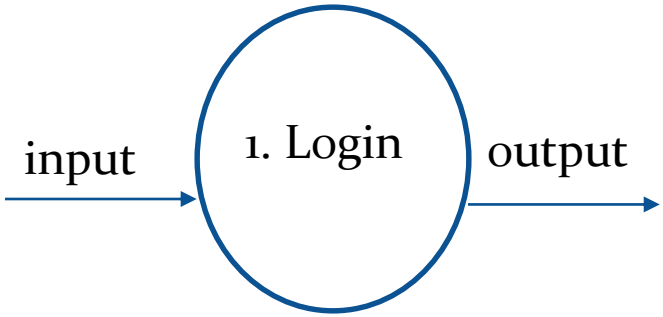
Can be user or external system that performs some process or activity in project

Symbolized with rectangle

Like, we have entity ‘admin’ then symbol will be

2)Process

- Work or action taken on incoming data to produce output
- Each process must have input and output
- Symbolized as



TOPS Technologies

3) Data Flow

- Can be used to show input and output of data
- Should be named uniquely and don’t include word ‘data’
- Names can be ‘payment’, ‘order’, ’complaint’ etc
- Symbolized as

TOPS Technologies

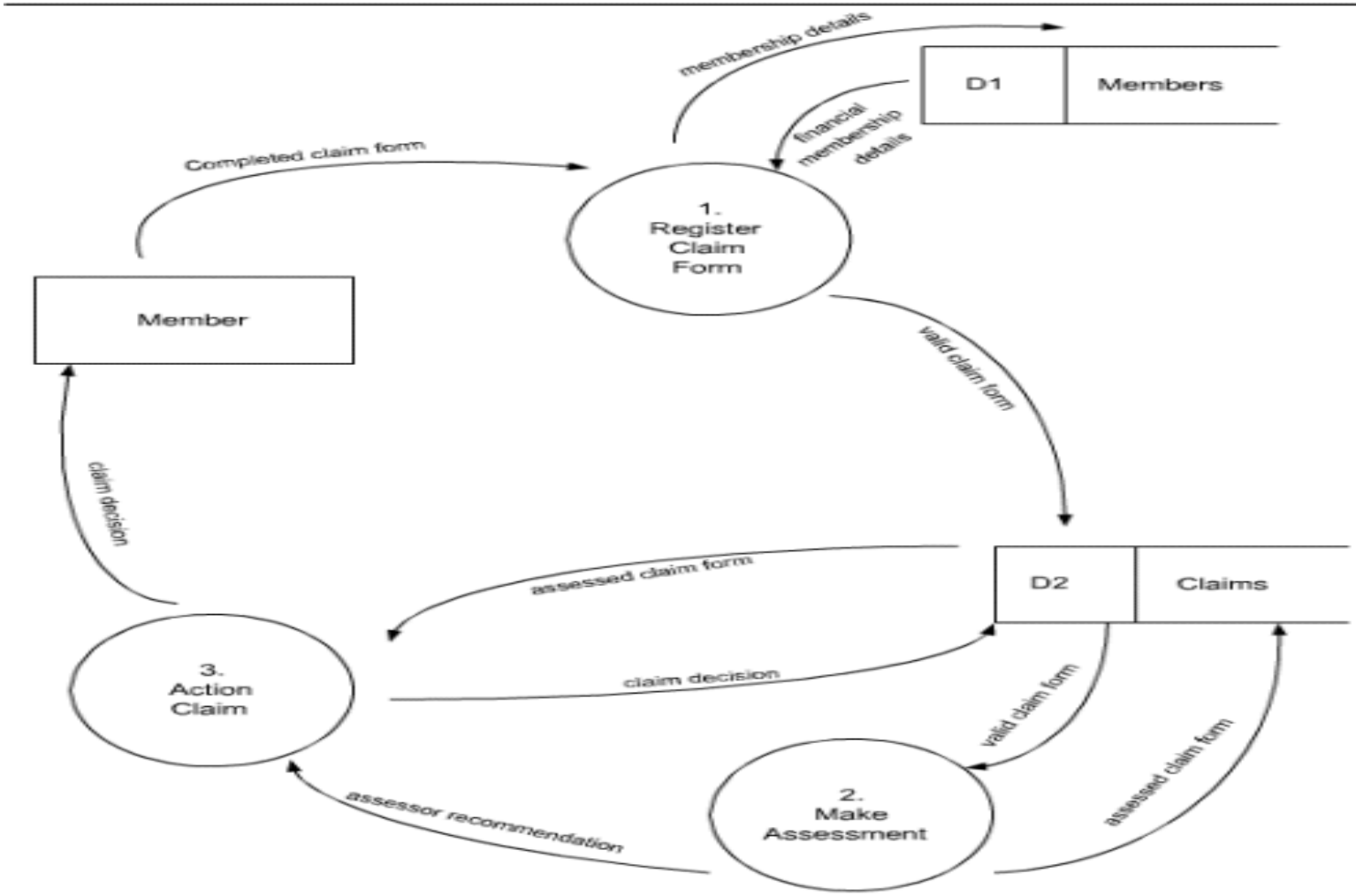
4) Data Store

- Can be used to show database tables
- Only process may connect data stores
- There can be two or more process sharing same data store
- Symbolized as

TOPS Technologies

DFD Rules

- 6 rules to follow
- Consider data flow diagram as given below



TOPS Technologies

Rule 1

Each process must have data flowing into it and coming out from it

Rule 2

Each data store must have data going inside and data coming outside

Rule 3

A data flow out of a process should have some relevance to one or more of the data flows into a process

In process 3 all data flows are connected to process claim.

The claim decision can not be made until the claim form has been submitted and the assessor makes a recommendation

Rule 4

Data stored in system must go through a process

Rule 5

Two data stores can’t communicate with each other unless process is involved in between

Rule 6

The Process in DFD must be linked to either another process or a data store

A process can’t be exist by itself, unconnected to rest of the system

Context Level

DFD level-0

It’s also context level DFD

Context level diagrams show all external entities.

They do not show any data stores.

The context diagram always has only one process labelled 0.

DFD Level-1(or 2)

- Include all entities and data stores that are directly connected by data flow to the one process you are breaking down show all other data stores that are shared by the processes in this breakdown
- Like login process will linked to users & database in further leveling

TOPS Technologies

Flow Charts…

- Used to show algorithm or process
- Can give step by step solution to the problem
- The first flow chart was made by John Von Newman in 1945
- Pictorial view of process
- Helpful for beginner and programmers

TOPS Technologies

- Flowcharts are generally drawn in the early stages of formulating computer solutions.
- Flowcharts facilitate communication between programmers and business people.
- These flowcharts play a vital role in the programming of a problem and are quite helpful in understanding the logic of complicated and lengthy problems.
- Once the flowchart is drawn, it becomes easy to write the program in any high level language.
- Often we see how flowcharts are helpful in explaining the program to others.
- Hence, it is correct to say that a flowchart is a must for the better documentation of a complex program.

Symbols are..

1)Start Or End

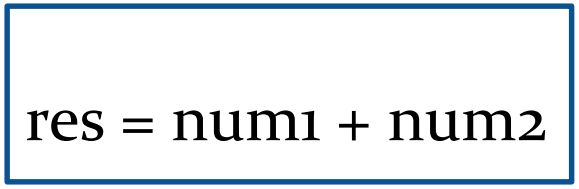
Show starting or ending of any flow chart

Symbolized as

OR

2. Process

- Defines a process like defining variables or initializing variable or performing any computation
- Symbolized as



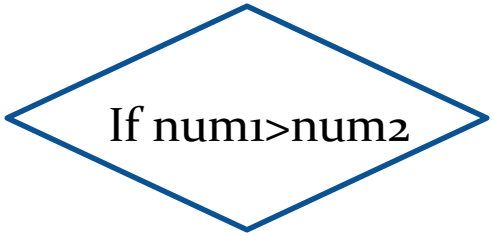
TOPS Technologies

3) Input or Output

- Used when user have to get or initialize any variable
- Like get num1 and num2 from user
- Symbolized as

4) Decision Making

- For checking condition this symbols can be used
- Like if num1 is greater than num2
- Can be symbolized as



5) Flow lines

- Lines showing flow of data and process
- Showing flow of instructions also
- Can be symbolized as

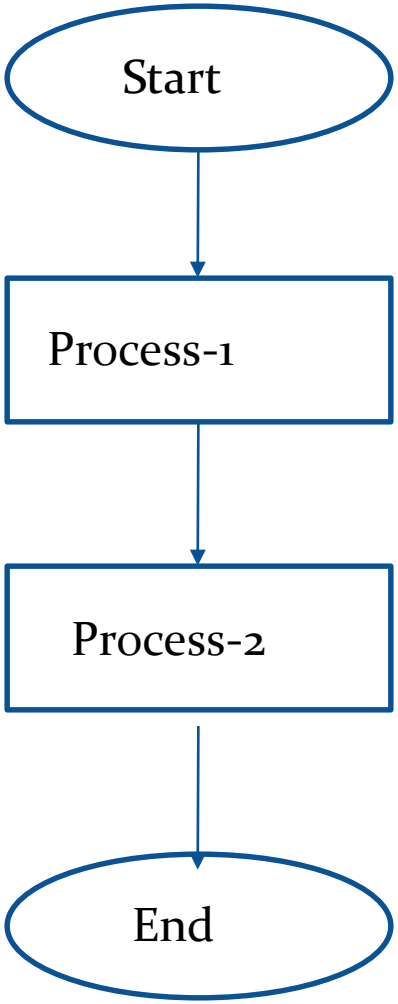
TOPS Technologies

Any program can be in three format

1. Linear or sequence
2. Branching
3. Looping

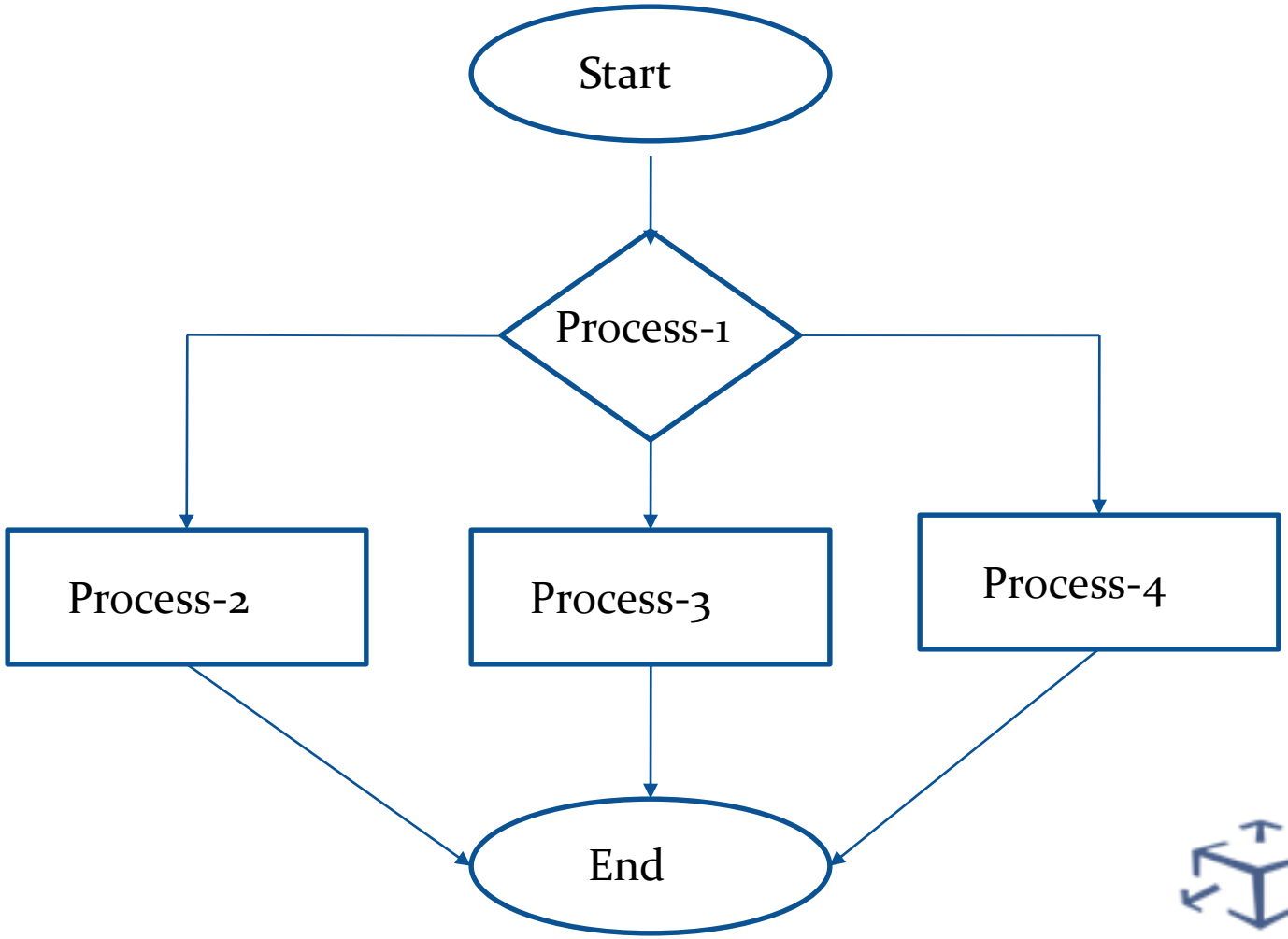
Following are notations can be used to show this format

Linear Program Structure



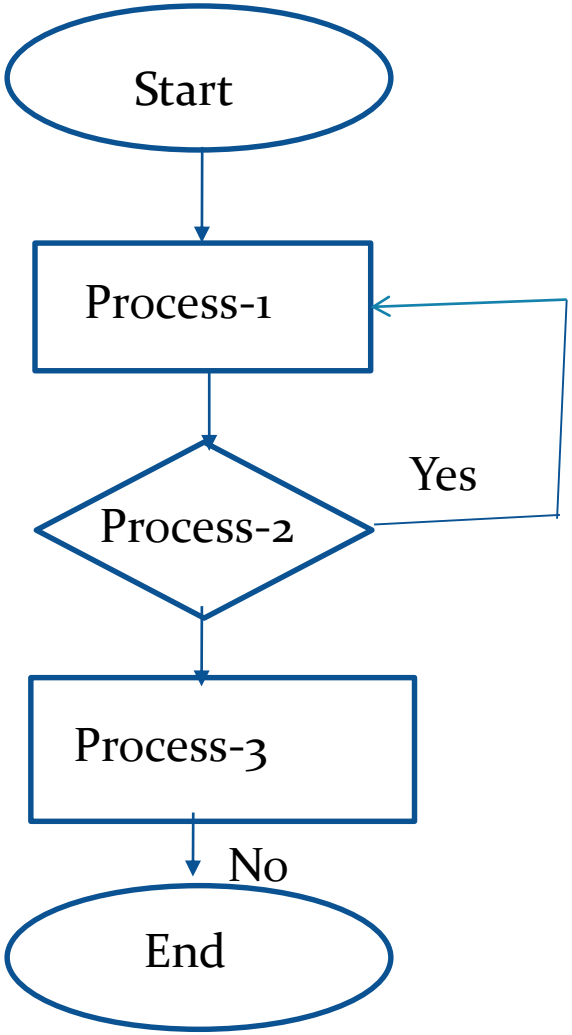
TOPS Technologies

Branching Program Structure..



TOPS Technologies

Looping Program Structure



TOPS Technologies

Module – 2[Core Java]

- Introduction
- JVM,JDK,JRE
- Source File Layout
- Data types
- Modifiers
- Conditional Statement and Looping Statement
- Arrays
- Encapsulation
- Inheritance
- Polymorphism
- Abstract and Interface Keywords(this,static,final,super)
- Classes
  - Object class,
  - String class,
  - wrapper class,
  - String buffer and String builder)

- Exceptions
- File I/O
- Collection Framework
- Thread
- JAVA GUI
- Layouts
- Events

TOPS Technologies

Introduction

- Java programming language was originally developed by Sun Microsystems.
- It is initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).
- Sun Microsystems was acquired by the Oracle Corporation.
- Now it is part of Oracle.
- Java is guaranteed to be Write Once, Run Anywhere.
- Platforms available :
  - J2SE for Standard Edition
  - J2EE for Enterprise Applications,
  - J2ME for Mobile Applications.

TOPS Technologies

- **Features of Java**
- Java is simple
- Java is object-oriented
- Java is distributed
- Java is interpreted
- Java is robust/powerful
- Java is secure
- Java is architecture-neutral
- Java is portable
- Java's performance
- Java is multithreaded
- Java is dynamic

TOPS Technologies

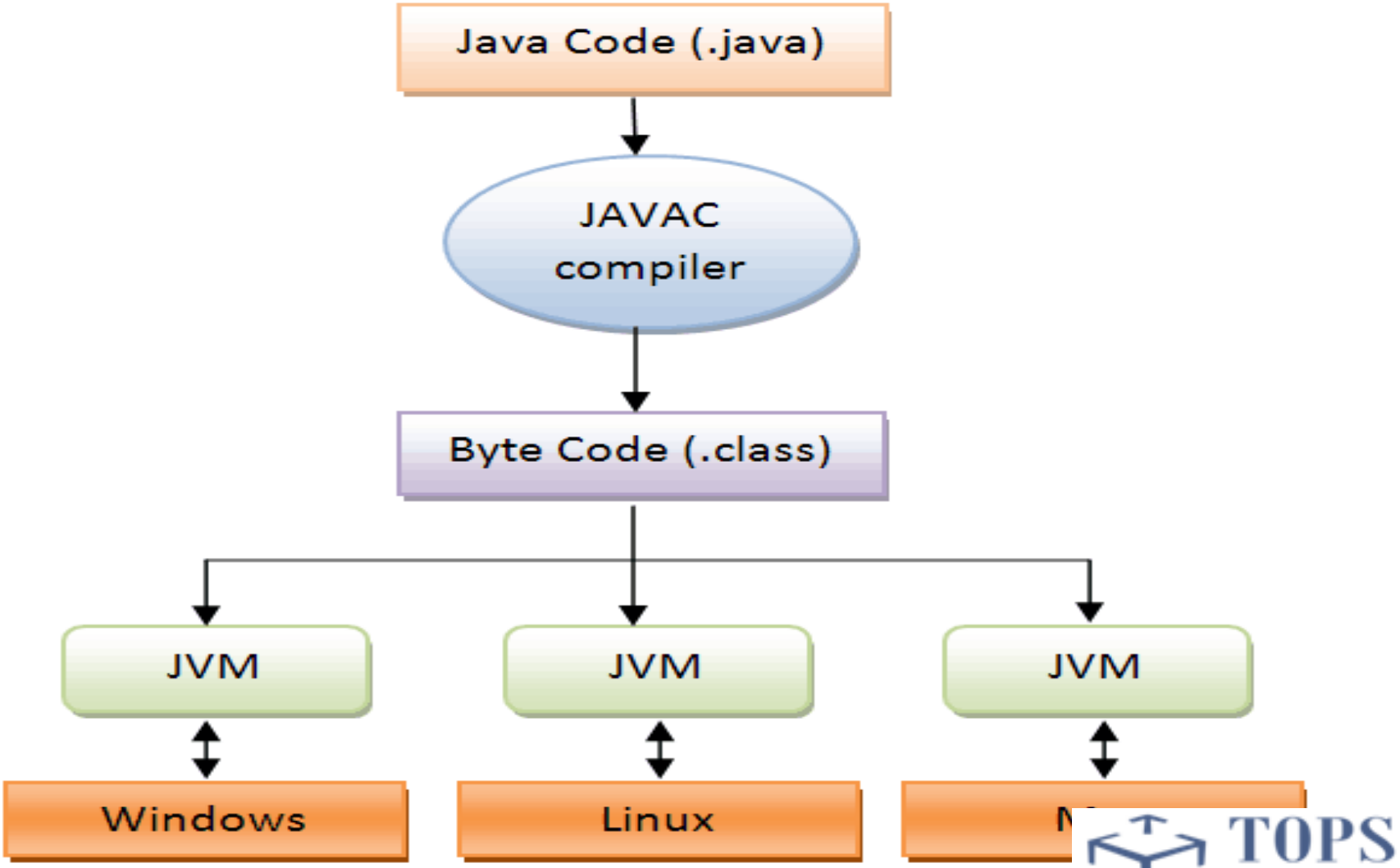
Advantages of JAVA

- It is an open source, so users do not have to struggle with heavy license fees each year
- Platform independent
- Java API's can easily be accessed by developers
- Java perform supports garbage collection, so memory management is automatic
- Java always allocates objects on the heap
- Java embraced the concept of exception specifications
- Multi-platform support language and support for web-services
- Using JAVA we can develop dynamic web applications
- It allows you to create modular programs and reusable codes

TOPS Technologies

Introduction of JVM , JDK and JRE

- **JVM** stands for **Java Virtual Machine**.
- It is the engine that drives the java code.
- It is the medium which compile java code to **bytecode** which get interpret on different machine and hence it makes it Platform/Operating system independent.
- Bytecode is an **intermediary language** between Java source and the host system.
- Java is called platform independent because of Java Virtual Machine.
- As different computers with different operating system have their own JVM, when we submit '**.class**' file to any operating system, JVM interpret the bytecode i[illegible] machine level language.


Java Code (.java)
JAVAC compiler
Byte Code (.class)
JVM
JVM
JVM
Windows
Linux

- **JDK (Java Development Kit)**

Java Development Kit contains the software and tools that you need to compile, debug, and run applets and applications that you've written using the Java programming language.

The JDK has as its primary components a collection of programming tools .

This tool also helps manage JAR files, javadoc .

The JDK also comes with a complete Java Runtime Environment, usually called a private runtime. It consists of a Java Virtual Machine and all of the class libraries present in the production environment, as well as additional libraries only useful to developers.

JDK versions

- JDK Alpha and Beta (1995)
- JDK 1.0 (January 23, 1996)
- JDK 1.1 (February 19, 1997)
- J2SE 1.2 (December 8, 1998)
- J2SE 1.3 (May 8, 2000)
- J2SE 1.4 (February 6, 2002)
- J2SE 5.0 (September 30, 2004)
- Java SE 6 (December 11, 2006)
- Java SE 7 (July 28, 2011)
- Java SE 8 (March 18, 2014)

TOPS Technologies

JRE

- The Java Runtime Environment (JRE), also known as Java Runtime, is part of the Java Development Kit (JDK),
- A set of programming tools for developing Java applications.
- The Java Runtime Environment provides the minimum requirements for executing a Java application.
- It consists of the Java Virtual Machine (JVM), core classes, and components, which are necessary to run programs written in Java.
- The JRE is available for multiple computer platforms, including Mac, Windows, and Unix.

TOPS Technologies

Java IDE Tools

- To write your Java programs, you will need a text editor. There are even more sophisticated IDEs available in the market.
- **Notepad:** On Windows machine you can use any simple text editor like Notepad, TextPad.
- **Netbeans:**is a Java IDE that is open-source.
- **Eclipse:** is also a Java IDE developed by the eclipse open-source community.

TOPS Technologies

About Eclipse IDE

- Most people know Eclipse as an integrated development environment (IDE) for Java.
- Today it is the leading development environment for Java with a market share of approximately 65%.
- The Eclipse IDE can be extended with additional software components.
- Eclipse calls these software components *plug-ins*. Several Open Source projects and companies have extended the Eclipse IDE

TOPS Technologies

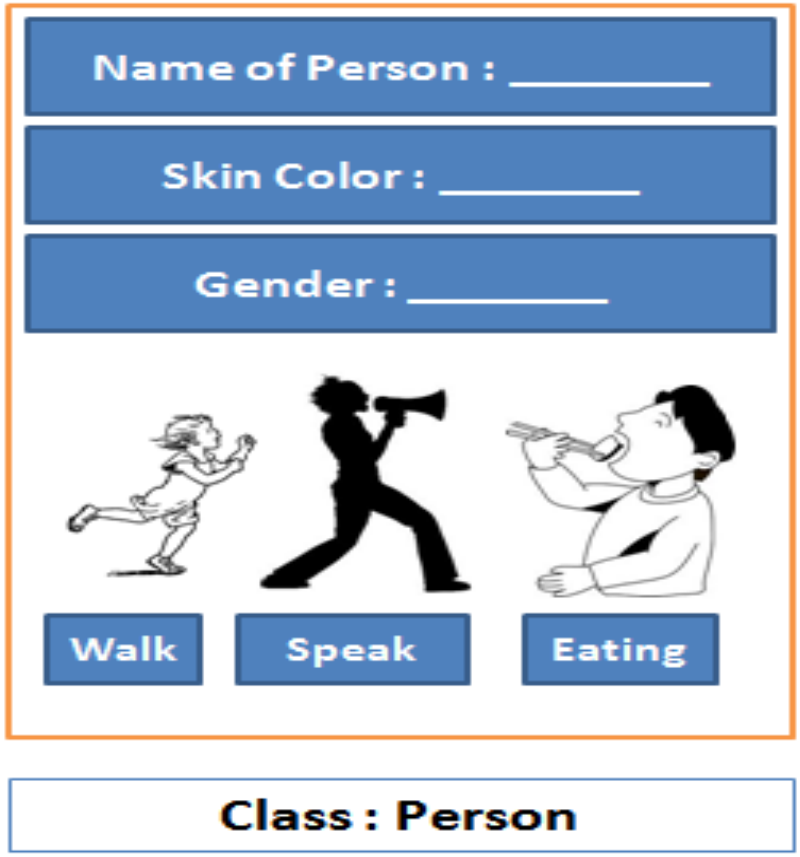
Introduction

- Class,Object , Class Members
- Constructor
- Garbase Collection
- Finallize

TOPS Technologies

Class Object and Members

- **Class**
  - A class can be defined as a template/blue print that describes the behaviors/states that object of its type support.



- Objects are key to understanding object-oriented technology.
- Examples of real-world objects:
  - your dog,
  - your desk,
  - your television set,
  - your bicycle.
- Real-world objects share two characteristics:
  - They all have state and behavior.
  - Example:
    - A dog has states - color, name, breed as well as behaviors - wagging, barking, eating.
    - An object is an instance of a class.

- An **object** is an instance of a class created using a new operator.
- The new operator returns a reference to a new instance of a class.
- Eg Person me=new Person();

Class Members

- When we create a class its totally incomplete without defining any member of this class.

- **Field:** field is nothing but the property of the class or object which we are going to create .
  - for example if we are creating a class called computer then they have property like model, mem_size, hd_size, os_type etc
- **Method:** method is nothing but the operation that an object can perform it define the behavior of object how an object can interact with outside world .
  - For example : startMethod (), shutdownMethod ().

Constructors

- 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.
- Each time a new object is created, at least one constructor will be invoked.
- The main rule of constructors is that they should have the same name as the class.
- A class can have more than one constructor.
- Constructor declarations look like method declarations—except that they use the name of the class and have no return type.

- They are called or invoked when an object of class is created and can't be called explicitly.
- E.g. public class Person{

```
//attribute declaration
Person(){
}
//method declaration  }
```

- Like other methods in java constructor can be overloaded i.e. we can create as many constructors in our class as desired.
- Number of constructors depends on the information about attributes of an object we have while creating objects.

TOPS Technologies

Java constructor chaining

- Constructor chaining occurs when a class inherits another class i.e. in inheritance, as in inheritance sub class inherits the properties of super class.
- i.e. we will see later (in super keyword)
- **Example :**
- https://github.com/TopsCode/Java/blob/master/Module-2/2.0%20Constructor/2.0.1%20conschaining.java

- **Inheritance with constructor**
- Class always call its parents class constructor first.

TOPS Technologies

Example :
https://github.com/TopsCode/Java/blob/master/Module-2/2.0%20Constructor/2.0.3%20inheritencewithconstructor.java

- **Copy Constructor**
- You can copy constructor
- **Example :**
- https://github.com/TopsCode/Java/blob/master/Module-2/2.0%20Constructor/2.0.2%20copyconstructor.java

Garbage Collection

- Objects are created on heap in Java irrespective of there scope e.g. local or member variable.
- Garbage collection is a mechanism provided by Java Virtual Machine to reclaim heap space from objects which are eligible for Garbage collection.
- Garbage Collection in Java is carried by a daemon thread called Garbage Collector.
- Before removing an object from memory Garbage collection thread invokes finalize () method of that object and gives an opportunity to perform any sort of cleanup required.
- you as Java programmer can not force Garbage collection in Java; it will only trigger if JVM thinks it needs a garbage collection based on Java heap size.

TOPS Technologies

finalize()

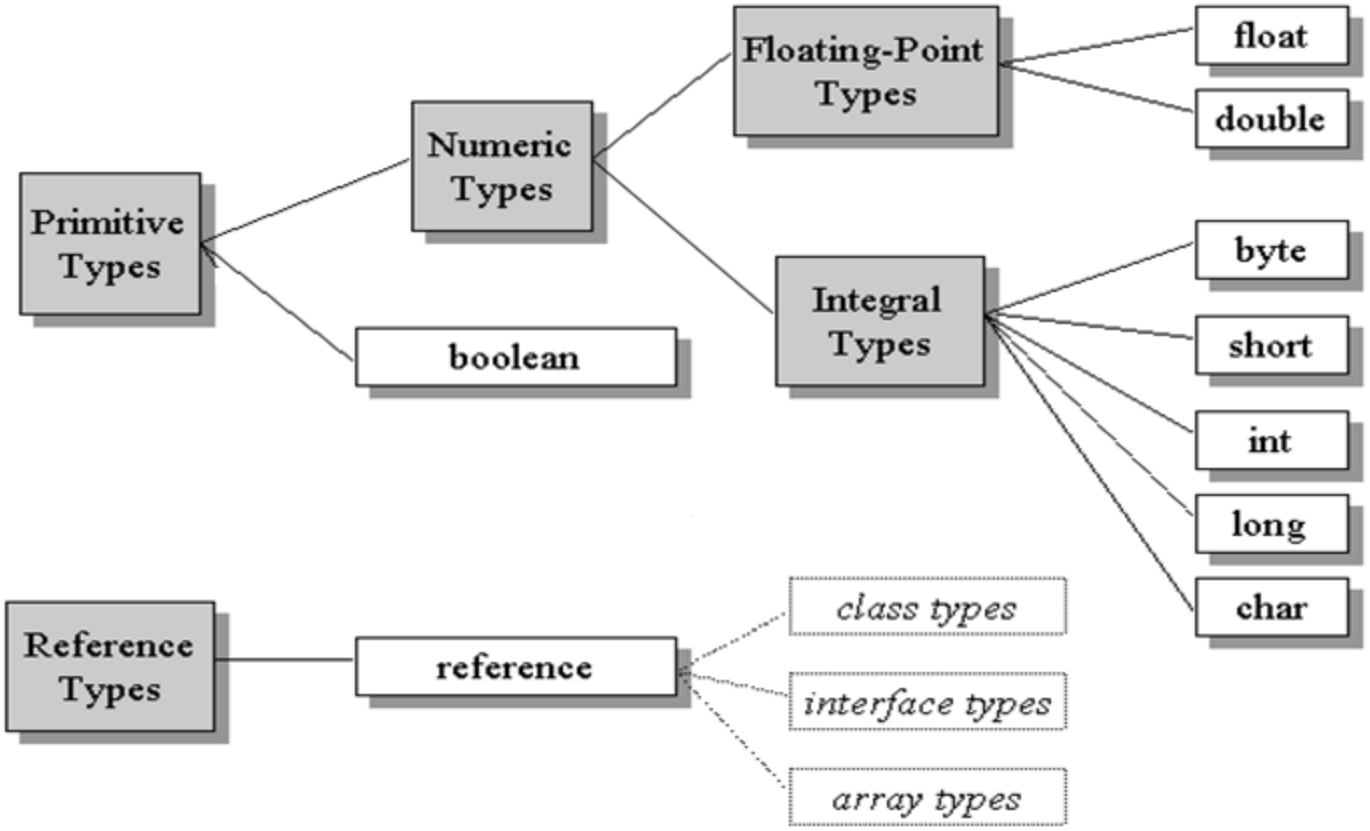
- There are methods like System.gc () and Runtime.gc () which is used to send request of Garbage collection to JVM but it's not guaranteed that garbage collection will happen.
- If there is no memory space for creating new object in Heap Java Virtual
  Machine throws OutOfMemoryError or java.lang.OutOfMemoryEr ror heap space.
- The java.lang.Object.finalize() is called by the garbage collector on an object when garbage collection determines that there are no more references to the object.
- A subclass overrides the finalize method to dispose of system resources or to perform other cleanup.

TOPS Technologies

Data types

- Variables are nothing but reserved memory locations to store values.
- Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.
- There are two data types available in Java:
  - Primitive Data Types
  - Reference/Object Data Types

TOPS Technologies



TOPS Technologies

Type Casting

Assigning a value of one type to a variable of another type is known as **Type Casting.**

Widening or Automatic type converion

Automatic Type casting take place when,

the two types are compatible

the target type is larger than the source type

- In Java a reference data type is a variable that can contain the reference or an address of dynamically created object.
- The reference data types are arrays, classes and interfaces.

class type

e.g.

```
public class Person{  …..
 Person p=new Person();//p is of class type variable.
 }
```

Array Type

- An array is a special kind of object that contains values called elements.
- The java array enables the user to store the values of the same type in contiguous memory allocations.
- E.g. int [] a = new int [10];
  String [] cities = {“Ahmedabad",“Mumbai", “Pune"};
- **Example :**
- https://github.com/TopsCode/Java/blob/master/Module-2/2.1%20Array/2.1.3%20ArrayDemo.java

Modifiers

- public,
- private,
- Protected,
- No modifier

- Modifiers are keywords that you add to those definitions to change their meanings.
- The Java language has a wide variety of modifiers, including the following:
- Java Access Modifiers (private ,protected and public)
- Non Access Modifiers (static ,final ,abstract, *synchronized* and *volatile*)
- In this session we see only Access Modifiers:
  - Visible to the package, the default. No modifiers are needed.
  - Visible to the class only (private).
  - Visible to the world (public).
  - Visible to the package and all subclasses (protected).

Modifier	Same Class	Same Package	Within Subclass outside the package	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

Conditional Statement and Looping Statement

- If,
- If ,else if,
- Switch..case
- For
- While
- Do …while

- There are two types of decision making statements in Java. They are:
  - if statements
  - switch statements.

The if Statement:

- An if statement consists of a Boolean expression followed by one or more statements.

Syntax:

```
if(Boolean_expression) {
//Statements will execute if the Boolean expression is true
}
```

- If the Boolean expression evaluates to true then the block of code inside the if statement will be executed.
- If not the first set of code after the end of the if statement (after the closing curly brace) will be executed.
- **Example :**
- https://github.com/TopsCode/Java/blob/master/Module-2/2.3%20Control%20Statment%20_%20Looping/2.3.1%20If

The if...else Statement:

- An if statement can be followed by an optional *else* statement, which executes when the Boolean expression is false.
- Syntax:

```
if(Boolean_expression){
//Executes when the Boolean expression is true
}
else{ //Executes when the Boolean expression is false
}
```

Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.3%20Control%20Statment%20_%20Looping/2.3.2%20IfElsePractical.java

The if...else if...else Statement:

- An if statement can be followed by an optional *else if...else* statement, which is very useful to test various conditions using single if...else if statement.

Syntax:

```
if(Boolean_expression 1){
//Executes when the Boolean expression 1 is true
}else if(Boolean_expression 2){
//Executes when the Boolean expression 2 is true }
else if(Boolean_expression 3){
//Executes when the Boolean expression 3 is true }
else { //Executes when the none of the above condition is true.
}
```

Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.3%20Control%20Statment%20_%20Looping/2.3.3%20IfElseLadderPractical.java

The switch Statement:

- A *switch* statement allows a variable to be tested for equality against a list of values.
- Each value is called a case, and the variable being switched on is checked for each case.
- Syntax:

  switch(expression){

  case value : //Statements break;

  //optional case value : //Statements break;

  //optional

  //You can have any number of case statements.

  default : //Optional //Statements }
- **Example :**
- https://github.com/TopsCode/Java/blob/master/Module-2/2.3%20Control%20Statment%20_%20Looping/2.3.3%20IfElseLadderPractical.java

Looping Statement

- Loops are specifically designed to perform repetitive tasks with one set of code.
- Loops save a lot of time.
- There are
  - while loop
  - do while loop &
  - for loop.

TOPS Technologies

The while loop:

Syntex

```
while ( <test_expr> )
<statement_or_block>
```

Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.3%20Control%20Statment%20_%20Looping/2.3.5%20WhileLoopPractical.java

The do/while loop:

Syntex:

```
do{
<statement_or_block>
}while ( <test_expr> );
```

Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.3%20Control%20Statment%20_%20Looping/2.3.6%20DoWhileLoop.java

TOPS Technologies

The for loop:

Syntex:

for (<init_expr>; <test_expr>; <alter_expr>)
<statement_or_block>

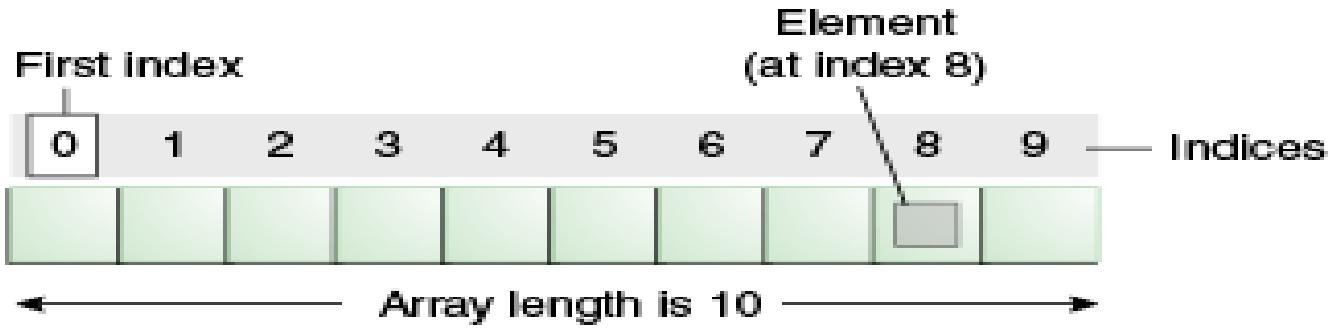
Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.3%20Control%20Statment%20_%20Looping/2.3.4%20ForLoopPractical.java

TOPS Technologies

Arrays

- **Introduction**
- An *array* is a container object that holds a fixed number of values of a single type.
- The length of an array is established when the array is created.
- After creation, its length is fixed.

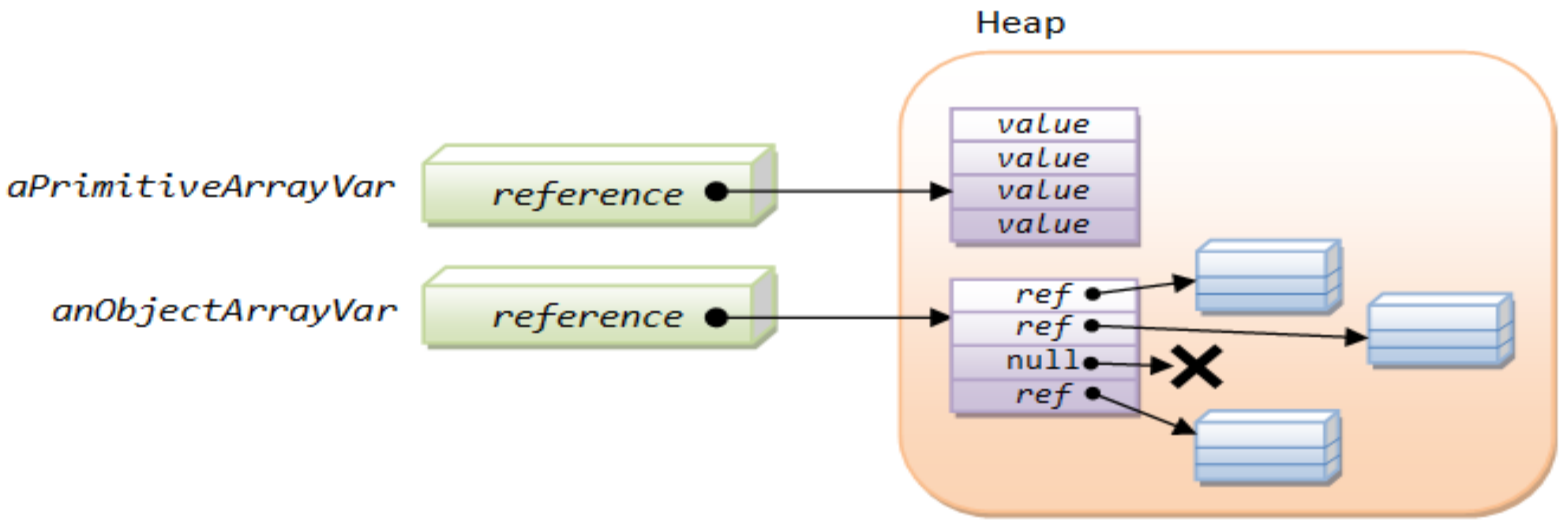


TOPS Technologies

Advantages of Array

- Arrays are most appropriate for storing a fixed amount of data which will be accessed in an unpredictable fashion.
- Arrays that has become very important on modern architectures is that iterating through an array has good locality of reference.
- Arrays are much faster than iterating through a linked list of the same size, which tends to jump around in memory.
- Arrays also are among the most compact data structures; storing 100 integers in an array takes only 100 times the space required to store an integer
- **Example :**
- https://github.com/TopsCode/Java/blob/master/Module-2/2.2%20Array/2.2.1%20ArrayBasic.java

TOPS Technologies



Array Types

There are 2 types of array available in java.

1. Primitive type,
    1. byte b; // byte is a primitive type
    2. byte[] arrayOfBytes; // byte[] is an array type: array of byte
    3. byte[][] arrayOfArrayOfBytes; // byte[][] is another type: array of byte[]
2. Class/Object type.
    1. Point[] points;

Resizing Array

- You can't resize an array in Java. You'd need to either:
  - Create a new array of the desired size, and copy the contents from the original array to the new array, using java.lang.System.arraycopy(...);
  - Use the java.util.ArrayList<T> class, which does this for you when you need to make the array bigger.(see later) .
  - Use java.util.Arrays.copyOf(...) methods which returns a bigger array, with the contents of the original array.

Coping Array

- The System class has an arraycopy() method that you can use to efficiently copy data from one array into another:
  - public static void arraycopy(Object src, int srcPos, Object dest, int destPos, int length)

Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.2%20Array/2.2.2%20ArrayPractical2.java

https://github.com/TopsCode/Java/blob/master/Module-2/2.2%20Array/2.2.3%20ArrayInputOutput.java

Reference type array:

Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.2%20Array/2.2.4%20ArrayWithObject.java

TOPS Technologies

Encapsulations

- Introduction
- Advantage of Encapsulations
- Example of Encapsulations

TOPS Technologies

Encapsulations

- Encapsulation is a practice to bind related functionality (Methods) & Data (Variables) in a protective wrapper (Class) with required access modifiers (public, private, default & protected) so that the code can be saved from unauthorized access by outer world and can be made easy to maintain.

TOPS Technologies

Advantages of Encapsulation in Java

- Encapsulation helps the developer to make the code more flexible and maintainable by binding related data in a single unit and access/restrict that using appropriate access modifier.
- With well encapsulation implementation, one can change one part of the code easily without affecting the other part of the code.
- **Example :**
- https://github.com/TopsCode/Java/blob/master/Module-2/2.4%20Oops/2.4.1%20Encapsulation/2.4.1.1%20EncapTest.java
- https://github.com/TopsCode/Java/blob/master/Module-2/2.4%20Oops/2.4.1%20Encapsulation/2.4.1.1%20Person.java

TOPS Technologies

Inheritance

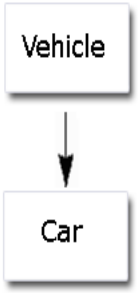
- Introduction
- Advantages of Inheritance
- Types of Inheritance
- Practical of Inheritance

TOPS Technologies

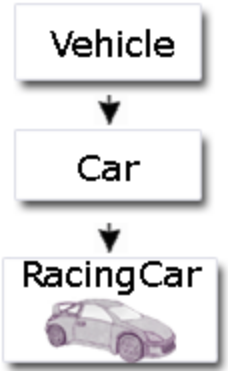
Inheritance

- Inheritance is a mechanism wherein a new class is derived from an existing class.
- In Java, classes may inherit or acquire the properties and methods of other classes.
- A class derived from another class is called a subclass, whereas the class from which a subclass is derived is called a super class.

- he following kinds of inheritance are there in java.
- **Simple Inheritance**



- **Multilevel Inheritance**



TOPS Technologies

Simple Inheritance

- When a subclass is derived simply from it's parent class then this mechanism is known as simple inheritance.
- In case of simple inheritance there is only a sub class and it's parent class.
- It is also called single inheritance or one level inheritance.
- **Example :**
- https://github.com/TopsCode/Java/blob/master/Module-2/2.4%20Oops/2.4.2%20Inheritance/2.4.1.1%20SingleInheritence.java

TOPS Technologies

Multilevel Inheritance

- It is the enhancement of the concept of inheritance. When a subclass is derived from a derived class then this mechanism is known as the multilevel inheritance.
- The derived class is called the subclass or child class for it's parent class and this parent class works as the child class for it's just above (parent) class.
- Multilevel inheritance can go up to any number of level.
- **Example :**
- https://github.com/TopsCode/Java/blob/master/Module-2/2.4%20Oops/2.4.2%20Inheritance/2.4.1.2%20MultilevelDemo.java

- Java does not support multiple Inheritance.
- Multiple Inheritance
- The mechanism of inheriting the features of more than one base class into a single class is known as multiple inheritance.
- Java does not support multiple inheritance but the multiple inheritance can be achieved by using the interface.
- In Java Multiple Inheritance can be achieved through use of Interfaces by implementing more than one interfaces in a class.

Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.4%20Oops/2.4.2%20Inheritance/2.4.1.3%20HierarchicalInheritance.java

https://github.com/TopsCode/Java/blob/master/Module-2/2.4%20Oops/2.4.2%20Inheritance/2.4.1.4%20inheritencewithconstructor.java

Polymorphism

- Types of Polymorphism
- Method Overloading and Overriding
- **Example :**
- https://github.com/TopsCode/Java/tree/master/Module-2/2.4%20Oops/2.4.3%20Polymorphism/2.4.3.3%20method_overriding

- https://github.com/TopsCode/Java/tree/master/Module-2/2.4%20Oops/2.4.3%20Polymorphism/2.4.3.4%20overloading_constructor

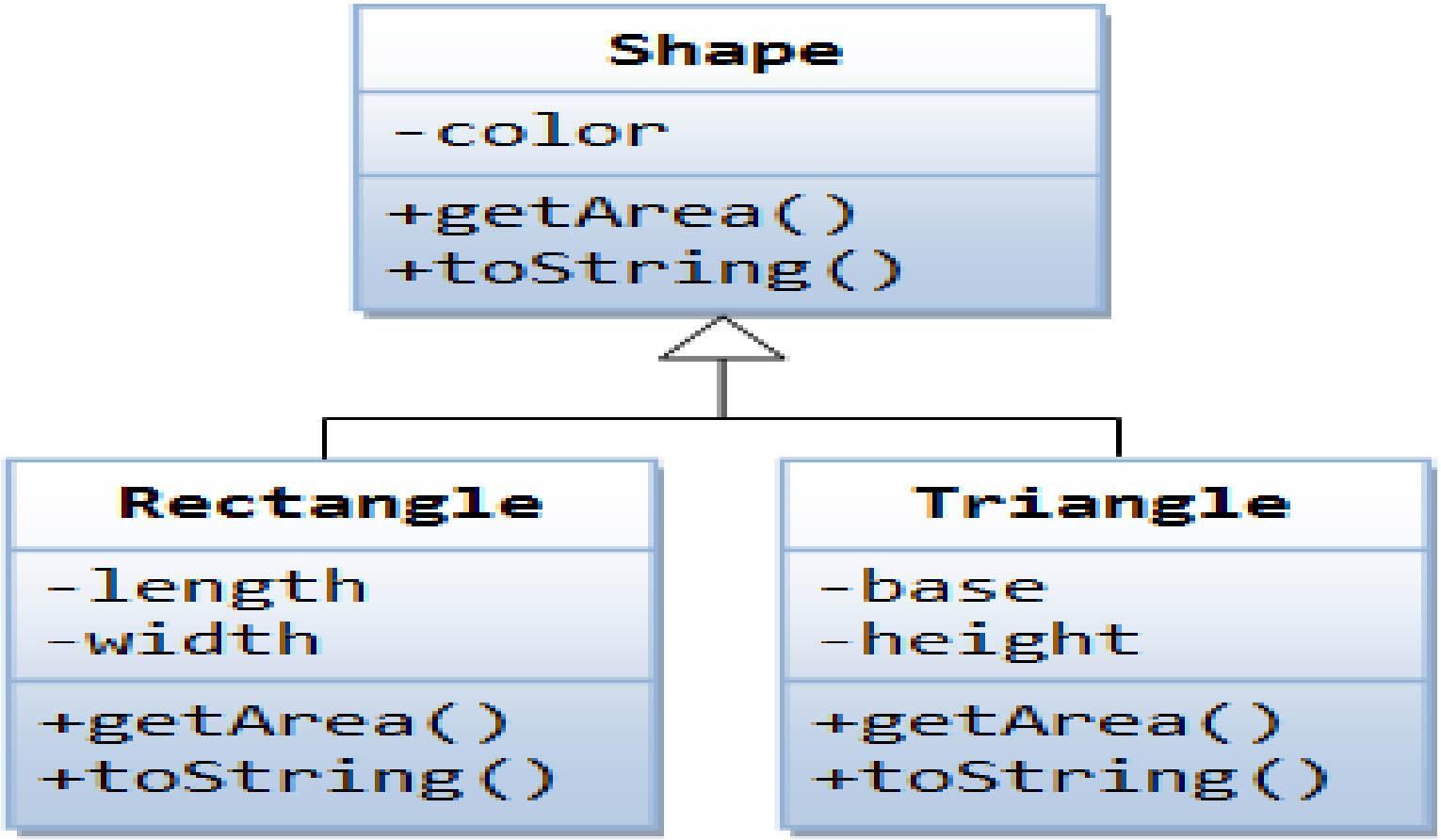
- https://github.com/TopsCode/Java/tree/master/Module-2/2.4%20Oops/2.4.3%20Polymorphism/2.4.3.3%20method_

TOPS Technologies

- Polymorphism is the ability of an object to take on many forms.
- Any Java object that can pass more than one IS-A test is considered to be polymorphic.
- **Example :**
- https://github.com/TopsCode/Java/blob/master/Module-2/2.4%20Oops/2.4.3%20Polymorphism/2.4.3.1%20Test.java
- Polymorphism is essentially considered into two versions.
  - Compile time polymorphism /static binding/method overloading.
  - Runtime polymorphism/dynamic binding/method overriding.

TOPS Technologies

Inheritance with Polymorphism



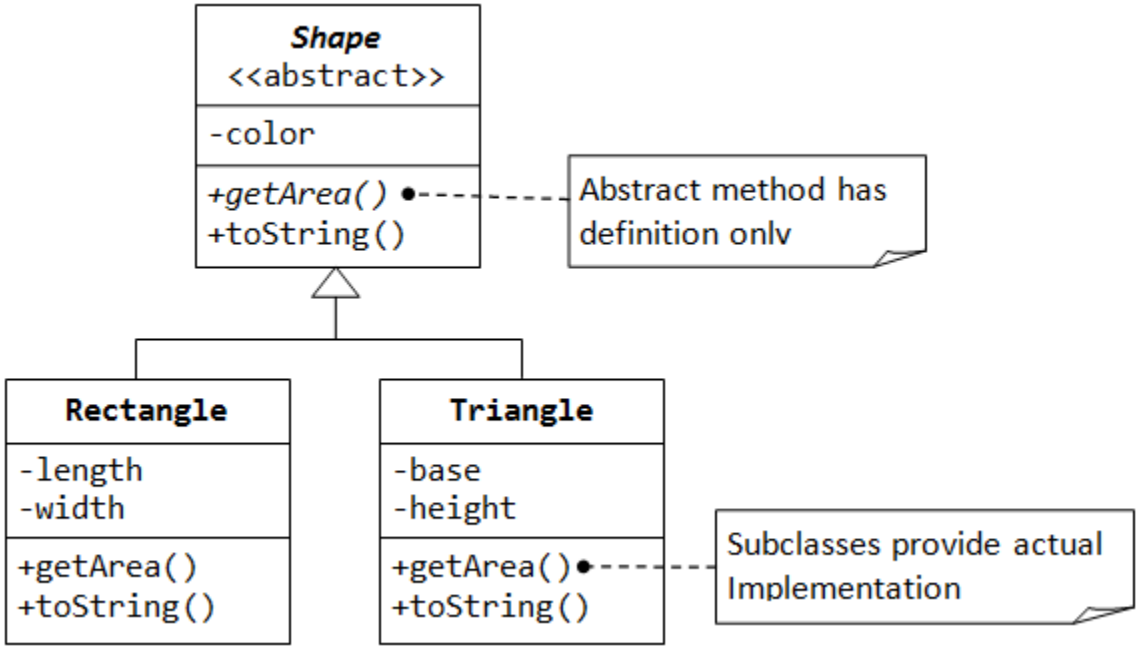
Abstract and Interface

Abstract

- Abstraction refers to the ability to make a class abstract in OOP.
- An abstract class is one that cannot be instantiated.
- If a class is abstract and cannot be instantiated, the class does not have much use unless it is subclass.
- An *abstract class* contains one or more *abstract methods*, which are simply method declarations without a body — that is, without executable code that implements the class or method.

TOPS Technologies

- An abstract method is like a prototype for a method, declaring the method's return type and parameter list but not providing an actual implementation of the method.



TOPS Technologies

Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.4%20Oops/2.4.4%20Abstraction/2.4.4.1%20abstractclass1.java

https://github.com/TopsCode/Java/blob/master/Module-2/2.4%20Oops/2.4.4%20Abstraction/2.4.4.2%20abstractclass2.java

- An interface is a collection of abstract methods.
- A class implements an interface, thereby inheriting the abstract methods of the interface.
- A class describes the attributes and behaviors of an object.
- An interface contains behaviors that a class implements.

Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.4%20Oops/2.4.4%20Abstraction/2.4.4.3%20InterfacePractical.java

class vs. interface
You cannot instantiate an interface.
An interface does not contain any constructors.
All of the methods in an interface are abstract.
An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.
An interface is not extended by a class; it is implemented by a class.
An interface is not extended by a class; it is implemented by a class.
An interface can extend multiple interfaces.

Keywords

- this,
- static,
- final,
- Super.

this

- ‘this’ keyword
- ‘this’ is reference variable that refers to the current object.
- Usage of ‘this’ ::
  - Refer current class instance variable.
  - Invoke current class constructor.
  - Invoke current class method.
  - Can be passed as argument in the method call.
  - Can be passed as argument in constructor call.
  - Used to return the current class instance.

Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.4%20Oops/ThisKeyWordPractical.txt

TOPS Technologies

super

- ‘super’ keyword.
- The ‘super’ is reference variable use to refer immediate parent class object.
- ‘super’ usage.
  - Super is used to refer immediate parent class instance variable.
  - Super() is used to invoke immediate parent class constructor.
  - Super is used to invoke immediate parent class method.
  - **Example :**

https://github.com/TopsCode/Java/blob/master/Module-2/2.4%20Oops/SuperPractical.txt

TOPS Technologies

static

- **‘static’ keyword.**
  - The *static* keyword is used as a modifier on
    - variables,
    - methods,
    - nested classes & blocks.
  - The *static* keyword declares the attribute or method is associated with the class as a whole rather than any particular instance of that class.
  - Thus static members are often called class members, such as class attributes or class methods.
  - **Example :**
  - https://github.com/TopsCode/Java/blob/master/Module-2/2.4%20Oops/StaticPractical.txt

final

final keyword

- ‘final’ m**o**difier can be used with
  - Class,
  - Variable &
  - Method.

final class

- This class can not extended.

final variable

- A final variable is a constant.
- A variable that is declared as final and not initialized is called a blank final variable.
- A blank final variable forces the constructors to initialise it.

final method

- You cannot override a final method.
- A blank final method variable must be set in the method body before being used.

Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.4%20Oops/FinalKeywordPractical.txt

Enumerations

- Enumerations was added to Java language in JDK5. **Enumeration** means a list of named constant. In Java, enumeration defines a class type. An Enumeration can have constructors, methods and instance variables. It is created using **enum** keyword. Each enumeration constant is *public*, *static* and *final* by default. Even though enumeration defines a class type and have constructors, you do not instantiate an **enum** using **new**. Enumeration variables are used and declared in much a same way as you do a primitive variable.

Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.5%20Enumeration/2.5.1%20Enum.java

TOPS Technologies

Object Class

- Class Object is the root of the class hierarchy.
- Every class has Object as a superclass.

Example :
https://github.com/TopsCode/Java/blob/master/Module-2/2.6%20Object%20Class/2.6.1%20ObjectClassPractical.java

- Some important methods:

Methods	Description
public String toString()	Returns a string representation of the object. In general, the toString method returns a string that "textually represents" this object.
void notify()	Wakes up a single thread that is waiting on this object's monitor.
void notifyAll()	Wakes up all threads that are waiting on this object's monitor.
void wait()	Causes the current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object.

==	Equals
equals() is a method defined inside the java.lang.Object class	== is one type of operator and you can compare both primitive and objects using equality operator in Java.
== is used to check reference or memory address of the objects whether they point to same location or not,	and equals() method is used to compare the contents of the object e.g. in case of comparing String its characters, in case of Integer its there numeric values etc. You can define your own equals method for domain object as per business rules e.g. two Employes objects are equal if there EmployeeId is same.
You can not change the behavior of == operator.	We can override equals() method and define the criteria for the objects equality.

String class

- The String class represents character strings.
- Strings are constant; their values cannot be changed after they are created.
- Some important methods.

Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.7%20String%20Class/2.7.1%20StringPractical.java

char **charAt**(int index)	Returns the char value at the specified index. An index ranges from 0 to length() - 1
public void **getChars**(int srcBegin, int srcEnd, char[] dst, int dstBegin)	Copies characters from this string into the destination character array.
public void **getBytes**(int srcBegin, int srcEnd, byte[] dst, int dstBegin)	Copies characters from this string into the destination byte array.

public boolean **equalsIgnoreCase**(String anotherString)	Compares this String to another String, ignoring case considerations. T
public String **substring**(int beginIndex)	Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.
public String concat(String str)	Concatenates the specified string to the end of this string.
public String **replace**(char oldChar, char newChar)	Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.
split public String[] **split**(String regex, int limit)	Splits this string around matches of the given regular expression.
public String **toLowerCase**()	Converts all of the characters in this String to lower case.

public String **toUpperCase**()	Converts all of the characters in this String to upper case
public String **trim**()	Returns a copy of the string, with leading and trailing whitespace omitted.
public char[] **toCharArray**()	Converts this string to a new character array.
public static String **valueOf**(char c)	Returns the string representation of the char argument.

StringBuffer & StringBuilder

Strings objects of type StringBuffer and Stringbuilder can be modified over and over again with out leaving behind a lot of new unused objects.

Example :

https://github.com/topscode/java/blob/master/module-2/2.7%20string%20class/2.7.2%20compare.java

StringBuffer	StringBuilder
StringBuffer methods are thread safe.	StringBuilders methods are not thread safe(not Synchronised).
Less efficient than StringBuilder.	More efficient than StringBuffer.
When the appliacation needs to run on multiple threads then it is better to use StringBuffer.	When the application needs to be run only in a single thread then it is better to use StringBuilder.

Wrapper Classes

- Each of Java's eight primitive data types has a class dedicated to it.
- These are known as wrapper classes.

Several possible reasons for wrapper classes availability,

- For the null value is possible,
- To include in a Collection,
- To treat generically / polymorphically as an Object along with other Objects

- Some Terms:
- **Boxing**
  - Boxing/wrapping, is the process of placing a primitive type within an object so that the primitive can be used as a reference object.
- **Autoboxing**
  - Autoboxing is the term for getting a reference type out of a value type just through type conversion (either implicit or explicit).
  - The compiler automatically supplies the extra source code which creates the object.
  - Integer i = new Integer(9);
  - Integer l = 9;

- **Unboxing**
  - Unboxing refers to getting the value which is associated to a given object, just through type conversion (either implicit or explicit).
  - The compiler automatically supplies the extra source code which retrieves the value out of that object, either by invoking some method on that object, or by other means.
  - Integer k = new Integer(4);
  - int l = k.intValue();
  - int m = k;

Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.8%20Wrapper%20Class/2.8.1%20WrapperPractical.java

TOPS Technologies

Exception

- Introduction
- Types of Exception
- try catch and finally block
- Multicatch Exceptions
- throw and throws keywords
- Method overriding with Exceptions
- Custom Exceptions

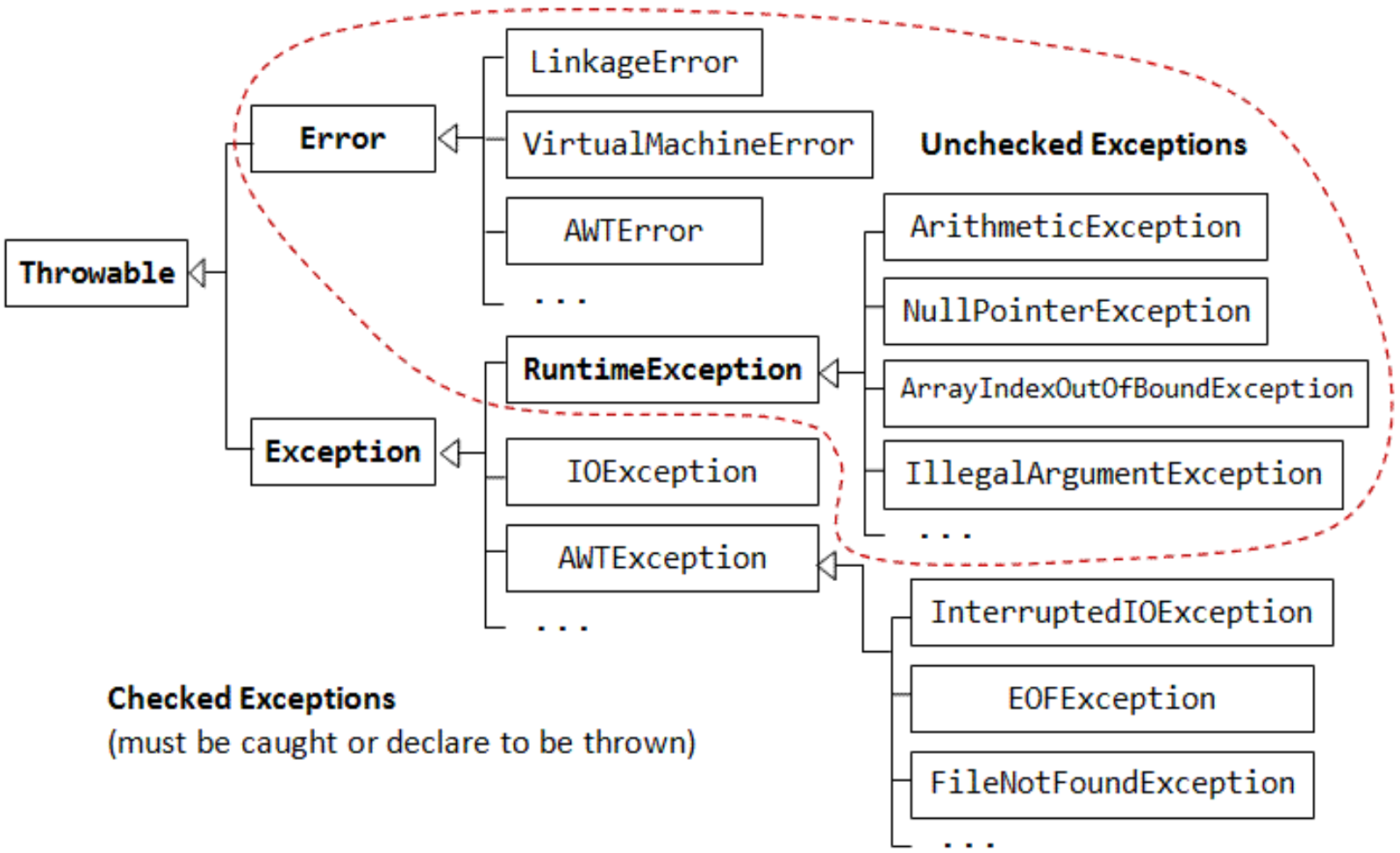
TOPS Technologies

Introduction

- An exception is a problem that arises during the execution of a program.
- An exception can occur for many different reasons:
  - A user has entered invalid data.
  - A file that needs to be opened cannot be found.
  - A network connection has been lost in the middle of communications or the JVM has run out of memory.

- There are three categories of exceptions:

Exception	Descrption
Checked exceptions:	A checked exception is an exception that is typically a user error or a problem that cannot be foreseen by the programmer. For example, if a file is to be opened, but the file cannot be found, an exception occurs. These exceptions cannot simply be ignored at the time of compilation.
Runtime exceptions:	A runtime exception is an exception that occurs that probably could have been avoided by the programmer. As opposed to checked exceptions, runtime exceptions are ignored at the time of compilation.
Errors:	These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. For example, if a stack overflow occurs, an error will arise. They are also ignored at the time of compilation.


Throwable
Error
LinkageError
VirtualMachineError
AWTError
...
Exception
RuntimeException
ArithmeticException
NullPointerException
ArrayIndexOutOfBoundException
IllegalArgumentException
...
IOException
AWTException
InterruptedIOException
EOFException
FileNotFoundException
...
Unchecked Exceptions
Checked Exceptions
(must be caught or declare to be thrown)

Exception Handling

try block

- The first step in constructing an exception handler is to enclose the code that might throw an exception within a try block.
- try
  - {
    - *code*
  - }
  - *catch and finally blocks . . .*

TOPS Technologies

catch block

- You associate exception handlers with a try block by providing one or more catch blocks directly after the try block.

  try {

  } catch (*ExceptionType name*) {

  }**Example :**

https://github.com/TopsCode/Java/blob/master/Module-2/2.9%20Exception/2.9.1%20TryCatchPractical.java

- **finally block**
- The finally keyword is used to create a block of code that follows a try block.
- **Example :**
- https://github.com/TopsCode/Java/blob/master/Module-2/2.9%20Exception/2.9.2%20MultiCatchFinally.java

TOPS Technologies

Multiple catch blocks

```
try
{ //Protected code }
catch(ExceptionType1 e1)
{ //Catch block }
catch(ExceptionType2 e2)
{ //Catch block }
catch(ExceptionType3 e3)
{ //Catch block }
```

- If an exception occurs in the protected code, the exception is thrown to the first catch block in the list.
- If the data type of the exception thrown matches ExceptionType1, it gets caught there.
- If not, the exception passes down to the second catch statement.

Example :

https://github.com/topscode/java/blob/master/module-2/2.9%20exception/2.9.2%20multicatchfinally.java

throw & throws keyword

- 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.
- You can throw an exception, either a newly instantiated one or an exception that you just caught, by using the **throw** keyword
- **Example :**
- https://github.com/TopsCode/Java/blob/master/Module-2/2.9%20Exception/2.9.3%20ThrowsPractical.java

TOPS Technologies

Custom Exception

- You can create your own exceptions in Java.
- For that,
  - All exceptions must be a child of Throwable.
  - If you want to write a checked exception that is automatically enforced by the Handle or Declare Rule, you need to extend the **Exception** class.
  - If you want to write a runtime exception, you need to extend the RuntimeException class.
  - Eg class MyException extends Exception{ }

Example :https://github.com/TopsCode/Java/blob/master/Module-2/2.9%20Exception/2.9.5%20BankDemo.java

Method overriding with Exception

- **Rules in methodoverriding**
- overriding method can not throw checked Exception which is higher in hierarchy, than checked Exception thrown by overridden method.
- For example if overridden method
  throws IOException or ClassNotfoundException, which
  are checked Exception, than overriding method can not
  throw java.lang.Exception because it comes higher in type hierarchy (it's super class of IOException and ClassNotFoundException).

Example :https://github.com/TopsCode/Java/blob/master/Module-2/2.9%20Exception/2.9.4%20ExceptionScope.java

File I/O

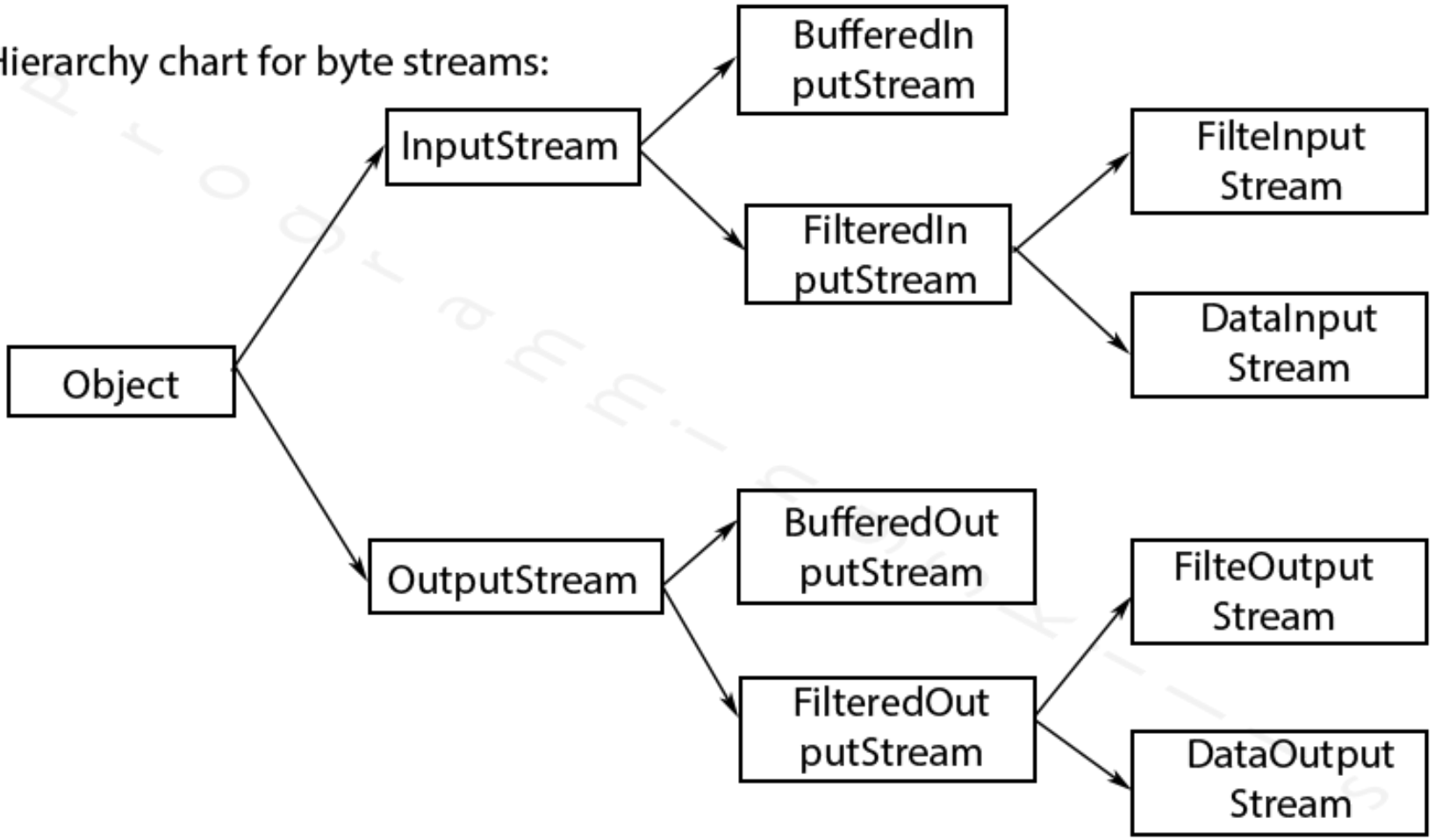
- What is Stream and Types of Stream
- File Input and Output Stream and its Methods
- File class
- Command Line Arguments.

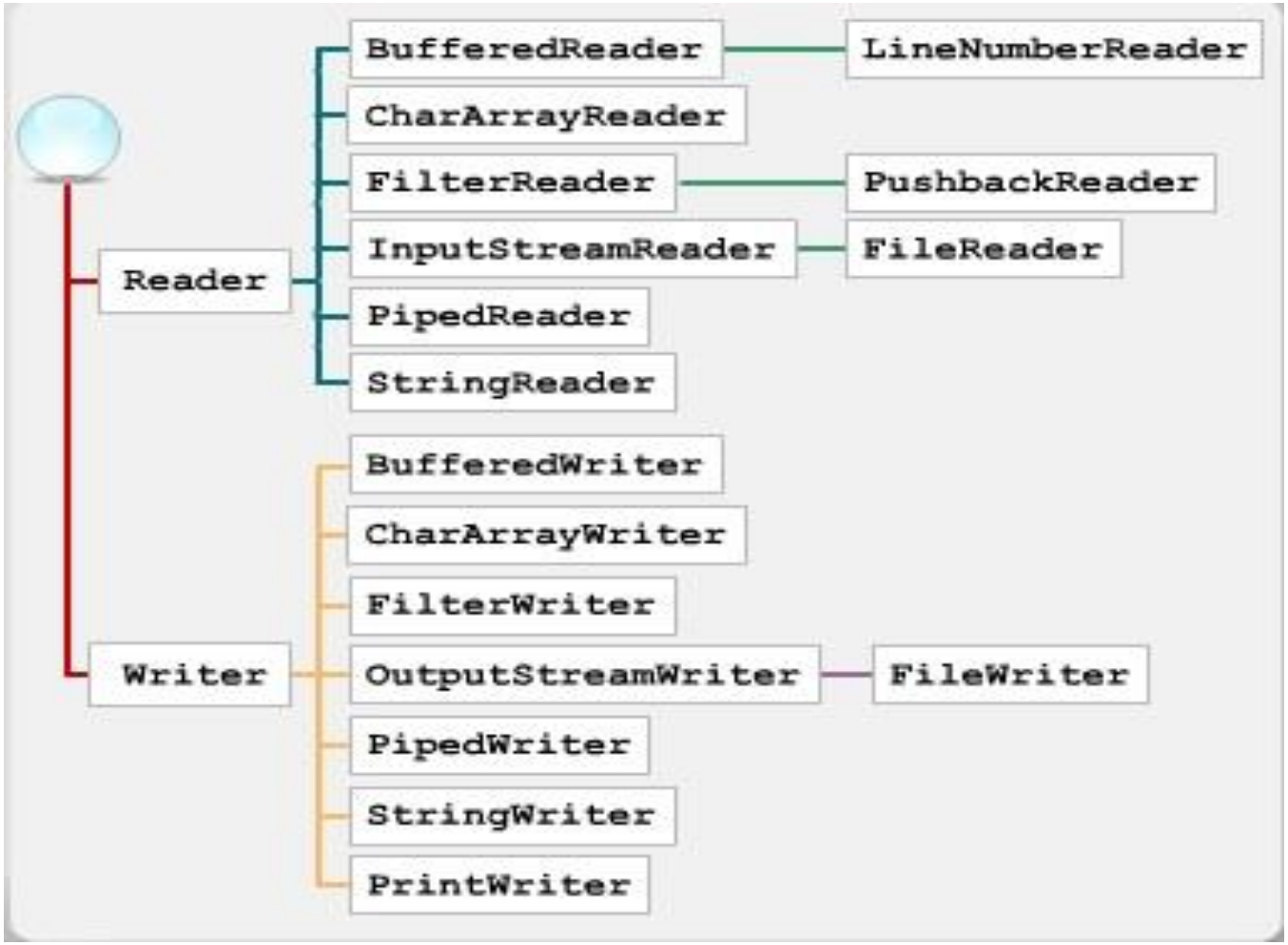
TOPS Technologies

- A stream can be defined as sequence of data from one place to another.
- A stream can represent many different kinds of sources and destinations, including disk files, devices, other programs, and memory arrays.
- Streams are used read/write data from file resides on network or hard disk or other storage device.
- An input stream/Source Stream initiates the flow of data, used to read data from Source.
- An output stream/Sink Stream terminates flow of data, used for writing data to a destination.

Example :
https://github.com/TopsCode/Java/blob/master/Module-2/2.10%20File%20IO/2.10.2%20FileInputOutput.java

Hierarchy chart for byte streams:




Reader
BufferedReader
LineNumberReader
CharArrayReader
FilterReader
PushbackReader
InputStreamReader
FileReader
PipedReader
StringReader
Writer
BufferedWriter
CharArrayWriter
FilterWriter
OutputStreamWriter
FileWriter
PipedWriter
StringWriter
PrintWriter

InputStream methods

- InputStream important methods.

Method details	
public int **available**() throws IOException	Returns the number of bytes that can be read (or skipped over) from this input stream without blocking by the next caller of a method for this input stream.
public void **close**() throws IOException	Closes this input stream and releases any system resources associated with the stream.
public void **mark**(int readlimit)	Marks the current position in this input stream.
public void **reset**() throws IOException	Repositions this stream to the position at the time the mark method was last called on this input stream.

Method details	
public boolean **markSupported**()	Tests if this input stream supports the mark and reset methods.
public abstract int **read**() throws IOException	Reads the next byte of data from the input stream. The value byte is returned as an int in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value -1 is returned.
public int **read**(byte[] b) throws IOException	Reads some number of bytes from the input stream and stores them into the buffer array b.
public int **read**(byte[] b, int off, int len) throws IOException	Reads up to len bytes of data from the input stream into an array of bytes. An attempt is made to read as many as len bytes.The number of bytes actually read is returned as an integer.

OutputStream methods

Method details	
public abstract void **write**(int b) throws IOException	Writes the specified byte to this output stream.
public void **write**(byte[] b) throws IOException	Writes b.length bytes from the specified byte array to this output stream. he general contract for write(b) is that it should have exactly the same effect as the call write(b, 0, b.length).
public void **write**(byte[] b, int off, int len) throws IOException	Writes len bytes from the specified byte array starting at offset off to this output stream.
public void **flush**() throws IOException	Flushes this output stream and forces any buffered output bytes to be written out.

Reader & Writer

- Java IO's Reader and Writer work much like the InputStream and OutputStream with the exception that Reader and Writer are character based.
- They are intended for reading and writing text.
- The InputStream and OutputStream were byte based.
- **Reader**
- The Reader is the baseclass of all Reader's in the Java IO API. Subclasses include a BufferedReader, PushbackReader etc.

Reader & Writer cont...

Methods	Description
abstract void close()	Closes the stream and releases any system resources associated with it.
void mark(int readAheadLimit)	Marks the present position in the stream.
boolean markSupported()	Tells whether this stream supports the mark() operation.
Int read()	Reads a single character.
Int read(char[] cbuf)	Reads characters into an array.
Int read(char[] cbuf,int off,int len)	Reads characters into a portion of an array.
Int read(CharBuffer targert)	Attempts to read characters into the specified character buffer.
void reset()	Resets the stream.
long skip()	Skips characters.

Reader & Writer cont...

- **Writer**
- You will normally use a Writer subclass rather than a Writer directly. Subclasses of Writer include OutputStreamWriter, CharArrayWriter, FileWriter, plus many others

Example :
https://github.com/TopsCode/Java/blob/master/Module-2/2.10%20File%20IO/2.10.3%20FileREaderWriter.java

TOPS Technologies

Reader & Writer cont...

Methods	Description
Writer append(char c)	Appends the specified character to this writer.
Writer append(CharSequence seq)	Appends the specified character sequence to this writer.
Writer append(CharSequence seq,int start,int end)	Appends a subsequence of the specified character sequence to this writer.
void close()	Closes the stream, flushing it first.
void flush()	Flushes the stream.
void write(char[] cbuf)	Writes an array of characters.
void write(char[] cbuf,int off,int len)	Writes a portion of an array of characters.
void write(int c)	Writes a single character.
void write(String str)	Writes a string.
Void write(String str,int off,int len)	Writes a portion of a string.

File Class

- Java File class represents the files and directory pathnames in an abstract manner.
- This class is used for creation of files and directories, file searching, file deletion etc.
- The File object represents the actual file/directory on the disk.
- There are following constructors to create a File object:
- File(File parent, String child);
- File(String pathname)
- File(String parent, String child)
- File(URI uri)

Example :https://github.com/TopsCode/Java/blob/master/Module-2/2.10%20File%20IO/2.10.1%20FileClassDEmo.java

TOPS Technologies

Method details	
public boolean **createNewFile**() throws IOException	Atomically creates a new, empty file named by this abstract pathname if and only if a file with this name does not yet exist.
public boolean **delete**()	Deletes the file or directory denoted by this abstract pathname. If this pathname denotes a directory, then the directory must be empty in order to be deleted.
public String[] **list**()	Returns an array of strings naming the files and directories in the directory denoted by this abstract pathname.
public boolean **mkdir**()	Creates the directory named by this abstract pathname.

Method details	
public long **length**()	Returns the length of the file denoted by this abstract pathname. The return value is unspecified if this pathname denotes a directory.
public boolean **isFile**()	Tests whether the file denoted by this abstract pathname is a normal file.
public boolean **isDirectory**()	Tests whether the file denoted by this abstract pathname is a directory.
public boolean **exists**()	Tests whether the file or directory denoted by this abstract pathname exists.
public boolean **canWrite**()	Tests whether the application can modify to the file denoted by this abstract pathname.
public boolean **canRead**()	Tests whether the application can read the file denoted by this abstract pathname.

Collection Framework

- Collection Framework Introducation
- Collection API
  - Arraylist
  - Hashset
  - Iterator
  - Hashmap
  - Vector and Enumration
- Generics
  - Genrics Example
- Comprator and Comprable.
- Vector vs Arraylist
- Hashmap vs Hashtable
- Comprator vs Comprable

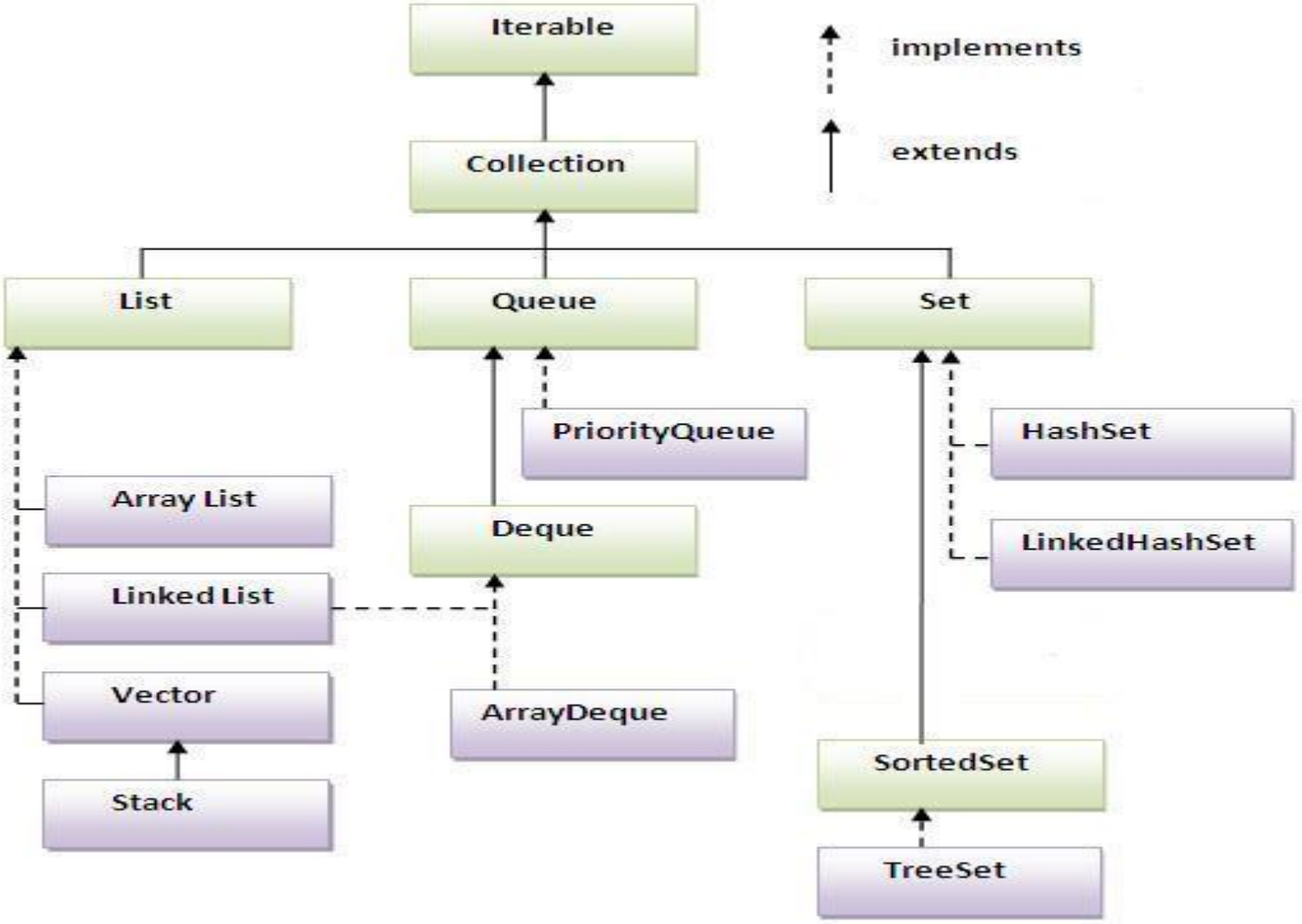
TOPS Technologies

Introduction

- A collection is a single object representing a group of objects known as its elements.
- The collections framework was designed to meet several goals.
  - The framework had to be high-performance. The implementations for the fundamental collections (dynamic arrays, linked lists, trees, and hash tables) are highly efficient.
  - The framework had to allow different types of collections to work in a similar manner and with a high degree of interoperability.
  - Extending and/or adapting a collection had to be easy.

- Towards this end, the entire collections framework is designed around a set of standard interfaces. Several standard implementations such as **LinkedList, HashSet,** and **TreeSet**, of these interfaces are provided that you may use as-is and you may also implement your own collection, if you choose.
- A collections framework is a unified architecture for representing and manipulating collections. All collections frameworks contain the following:
  - **Interfaces:** These are abstract data types that represent collections. Interfaces allow collections to be manipulated independently of the details of their representation. In object-oriented languages, interfaces generally form a hierarchy.
  - **Implementations, i.e., Classes:** These are the concrete implementations of the collection interfaces. In essence, they are reusable data structures.
  - **Algorithms:** These are the methods that perform useful computations, such as searching and sorting, on objects that implement collection interfaces. The algorithms are said to be polymorphic: that is, the same method can be used on many different implementations of the appropriate collection interface.
- In addition to collections, the framework defines several map interfaces and classes. Maps store key/value pairs. Although maps are not *collections* in the proper use of the term, but they are fully integrated with collections.

Collection API



ArrayList

- The ArrayList class extends AbstractList and implements the List interface. ArrayList supports dynamic arrays that can grow as needed.
- Standard Java arrays are of a fixed length. After arrays are created, they cannot grow or shrink, which means that you must know in advance how many elements an array will hold.
- Array lists are created with an initial size. When this size is exceeded, the collection is automatically enlarged. When objects are removed, the array may be shrunk.

Example :https://github.com/TopsCode/Java/blob/master/Module-2/2.11%20Collection/2.11.2%20ArrayListDemo.java

HashSet

- HashSet extends AbstractSet and implements the Set interface. It creates a collection that uses a hash table for storage.
- A hash table stores information by using a mechanism called hashing. In hashing, the informational content of a key is used to determine a unique value, called its hash code.
- The hash code is then used as the index at which the data associated with the key is stored. The transformation of the key into its hash code is performed automatically.

Example :https://github.com/TopsCode/Java/blob/master/Module-2/2.11%20Collection/2.11.6%20HashSetDemo.java

Iterator Interface

- Iterator interface will be use for the fetching the element from the collections.
- The easiest way to do this is to employ an iterator, which is an object that implements either the Iterator or the ListIterator interface.
- Iterator enables you to cycle through a collection, obtaining or removing elements. ListIterator extends Iterator to allow bidirectional traversal of a list, and the modification of elements.
- Before you can access a collection through an iterator, you must obtain one. Each of the collection classes provides an iterator( ) method that returns an iterator to the start of the collection.
- By using this iterator object, you can access each element in the collection, one element at a time.
- In general, to use an iterator to cycle through the contents of a collection, follow these steps:
  - Obtain an iterator to the start of the collection by calling the collection's iterator( ) method.
  - Set up a loop that makes a call to hasNext( ). Have the loop iterate as long as hasNext( ) returns true.
  - Within the loop, obtain each element by calling next( ).

Example :
https://github.com/TopsCode/Java/blob/master/Module-2/2.11%20Collection/2.11.1%20ArraListDemo2.java

HashMap

- The HashMap class uses a hashtable to implement the Map interface. This allows the execution time of basic operations, such as get( ) and put( ), to remain constant even for large sets.
  Example :https://github.com/TopsCode/Java/blob/master/Module-2/2.11%20Collection/2.11.10%20HashMapDemo.java

- **Stack**
- Stack
- The stack is a class.
- Which is used to represent LIFO data structure.
- There are push and pop method for data insertion and deletion.
- **Example** https://github.com/TopsCode/Java/blob/master/Module-2/2.11%20Collection/2.11.4%20StackDemo.java

TOPS Technologies

Vector and Enumeration Example

Example :

https://github.com/topscode/java/blob/master/module-2/2.11%20collection/2.11.5%20vectordemo.java

TOPS Technologies

Generics

- 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, respectively.
- Generics also provide compile-time type safety that allows programmers to catch invalid types at compile time.
- Using Java Generic concept, we might write a generic method for sorting an array of objects, then invoke the generic method with Integer arrays, Double arrays, String arrays and so on, to sort the array elements.

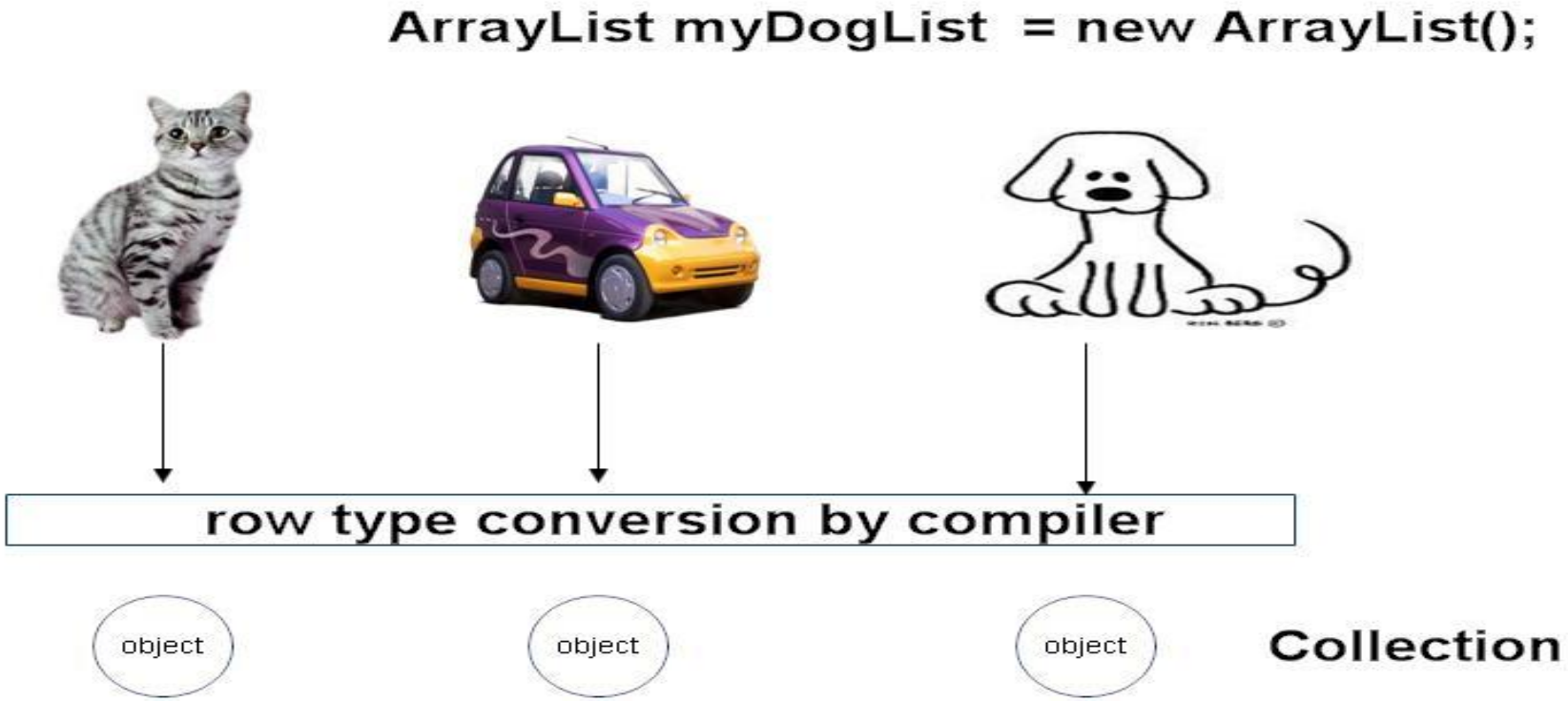
Generics Method

- You can write a single generic method declaration that can be called with arguments of different types. Based on the types of the arguments passed to the generic method, the compiler handles each method call appropriately. Following are the rules to define Generic Methods:
  - All generic method declarations have a type parameter section delimited by angle brackets (< and >) that precedes the method's return type (< E > in the next example).
  - Each type parameter section contains one or more type parameters separated by commas. A type parameter, also known as a type variable, is an identifier that specifies a generic type name.
  - The type parameters can be used to declare the return type and act as placeholders for the types of the arguments passed to the generic method, which are known as actual type arguments.
  - A generic method's body is declared like that of any other method. Note that type parameters can represent only reference types, not primitive types (like int, double and char).

Example :

https://github.com/topscode/java/blob/master/module-2/2.11%20collection/2.11.10%20hashmapdemo.java

Collection Framework Without Generic



Comparator and Comporable

- **Comparator** is capable of comparing two different objects. The method required for implementation is *compare()*.

- **Comparable** is implemented by a class in order to be able to comparing object of itself with some other objects. The class itself must implement the interface in order to be able to compare its instance(s). The method required for implementation is *compareTo()*.

Example :
https://github.com/TopsCode/Java/blob/master/Module-2/2.11%20Collection/2.11.11%20ComparableTEst.java

https://github.com/TopsCode/Java/blob/master/Module-2/2.11%20Collection/2.11.12%20ComparatorTest.java

Thread

- Introduction
- Thread Life Cycle.
- Creating Thread
- Thread class Methods
- Runnable Interface
- Synchronized block and Synchronized method.

TOPS Technologies

Thread Introduction

- **Introduction to Thread**
- A thread is an independent path of execution within a program.
- Many threads can run concurrently within a program.
- Every thread in Java is created and controlled by the **java.lang.Thread class**.
- A Java program can have many threads, and these threads can run concurrently, either asynchronously or synchronously.

Thread Introduction

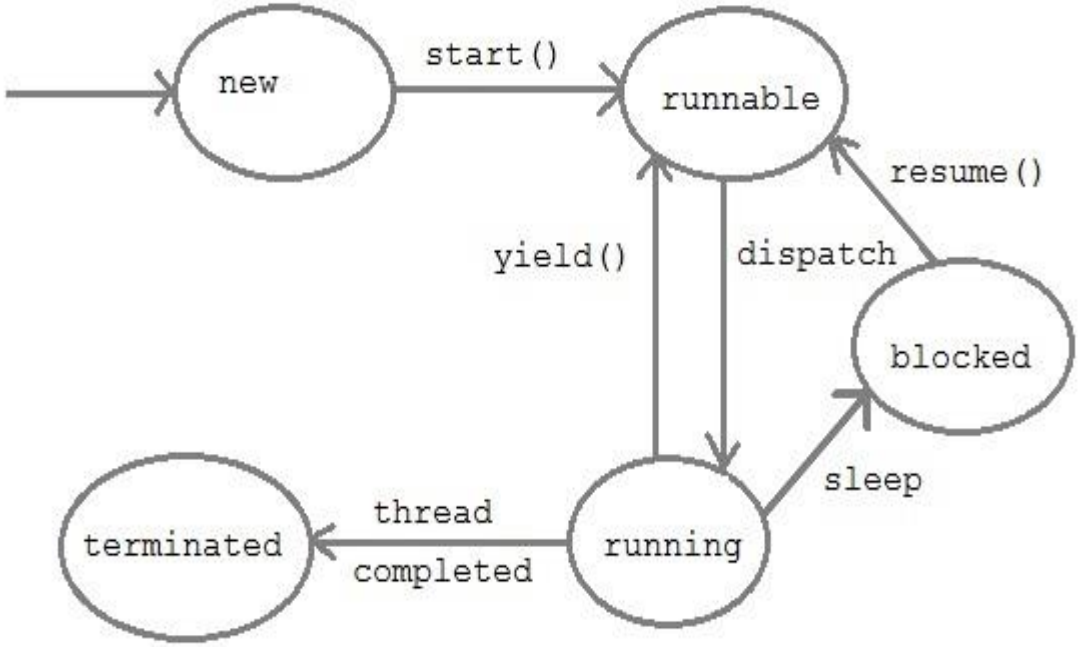
- Multithreading refers to two or more tasks executing concurrently within a single program.
- Multithreading has several advantages over Multiprocessing such as;
- Threads are lightweight compared to processes
- Threads share the same address space and therefore can share both data and code
- Context switching between threads is usually less expensive than between processes
- Cost of thread intercommunication is relatively low that that of process intercommunication
- Threads allow different tasks to be performed concurrently.

Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.12%20Thread/2.12.1%20MyThread.java

Thread Life cycle

- **Thread l**



TOPS Technologies

Thread Life cycle

Newborn State	When we create a thread object, the thread is born and is said to be in newborn state. The thread is not yet scheduled for running.
	Schedule it for running using **start()** method.
Runnable State :	The runnable state means that the thread is ready for execution and is waiting for the availability of the processor.
	That is the thread has joined the queue of threads that are waiting for execution.
	If all threads have equal priority, then they are given time slots for execution in round robin fashion, i.e. first-come, first-serve manner. The thread that relinquishes control joins the queue at the end and again waits for its turn.

	However, if we cant a thread to relinquish /leaving control to another thread of equal priority before its turn comes, we can do so by using the **yield()** method.
Running State :	Running means that the processor has given its time to the thread for its execution.
	It has been suspended using **suspend()** method. A suspended thread can be revived by using the **resume()** method.
	It has been made to sleep, we can put a thread to sleep for a specified time period using the method **sleep(time)** where time is in milliseconds.
	It has been told to wait until some event occurs. This is done using the**wait()** method. The thread can be scheduled to run again using the **notify()**method.

Blocked State :	A thread is said to be blocked when it is prevented from entering into the runnable state and subsequently the running state. this happens when the thread is suspended, sleeping, or waiting in order to satisfy certain requirements.
Dead State :	Every thread has a life cycle. A running thread ends its life when it has completed executing its **run()** method.

Thread Creation

- **Thread Creation**
- Thread can be created in 2 ways.
  - Extend the java.lang.Thread class.
  - **Example :**

https://github.com/TopsCode/Java/blob/master/Module-2/2.12%20Thread/2.12.2%20ThreadWithClass.java

  - Implement the Runnable interface.

Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.12%20Thread/2.12.3%20ThreadWithInterface.java

Thread methods

Method detail	
currentThread() :	Returns reference to the currently executing thread object.
getName() :	Returns the name of the thread in which it is called.
getPriority() :	Returns the Thread's priority
interrupt() :	Used for Interrupting the thread.
isAlive()	Used for testing whether a thread is alive or not.
setName()	Changes the name of the thread to NewName.
setPriority()	Changes the priority of thread.
sleep()	Causes the currently executing thread to sleep for the specified number of microsecond.
start() :	Used to begin execution of thread. The java virtual machine calls the run method of the thread in which this method is called.

Runnable Interface

- A Thread can be created by extending Thread class also. But Java allows only one class to extend, it wont allow multiple inheritance.
- So it is always better to create a thread by implementing Runnable interface. Java allows you to implement multiple interfaces at a time.
- By implementing Runnable interface, you need to provide implementation for run() method.
- To run this implementation class, create a Thread object, pass Runnable implementation class object to its constructor.
- Call start() method on thread class to start executing run() method.

Synchronization

- At times when more than one thread try to access a shared resource, we need to ensure that resource will be used by only one thread at a time. The process by which this is achieved is called **synchronization. The synchronization keyword in java creates a block of code referred to as critical section.**Every Java object with a critical section of code gets a lock associated with the object. To enter critical section a thread need to obtain the corresponding object's lock.
- **Why we use Syncronization ?**
- Consider an example, Suppose we have two different threads **T1 and T2, T1 starts execution and save certain values in a file *temporary.txt which will be used to calculate some result when T1 returns. Meanwhile, T2 starts and before T1 returns, T2 change the values saved by T1 in the file temporary.txt (temporary.txt is the shared resource). Now obviously T1 will return wrong result.***

- **Using Synchronized Methods**
- Using Synchronized methods is a way to accomplish synchronization. But lets first see what happens when we do not use synchronization in our program.

JAVA

Release Dates

Java 9-27 July,2017

Java 10-20 March,2018

JAVA

Java 9 Updates

- Private method in Interface
- REPL(Read-Evaluate-Print-Loop) JShell
- Collection Factory Methods
- Anonymous Inner Class

JAVA

Private method in Interface

- Used to avoid data redundancy(duplication of code)

- In Java 8, an interface can have only four kinds of things:
    1. Constant variables
    2. Abstract methods
    3. Default methods
    4. Static methods

- In Java 9 and later versions, an interface can have six kinds of things:

    1. Constant variables
    2. Abstract methods
    3. Default methods
    4. Static methods
    5. Private methods
    6. Private Static methods

TOPS Technologies

JAVA

Rules for private method in interface.

- No private and abstract modifiers together, it will give compiler error.
  - The “private” method means fully implemented method because sub-classes cannot inherit and override this method.
  - The “abstract” method means no-implementation method. Here sub-classes should inherit and override this method.

- Private methods must contain body.

TOPS Technologies

JAVA

Java 9 REPL(Read-Evaluate-Print-Loop) JShell

- Command line interactive tool.
- We can code and test java based logic without compiling using javac and see the result of calculations directly.
- Without writing import, package statement and whole class code we can test part of code in JShell.

JAVA

1) Running Jshell

-Open command prompt and type jshell.

```
$ jshell|  Welcome to JShell -- Version 9-ea|  For
an introduction type: /help introjshell>
```

2) Exiting Jshell

-Type /exit.

```
jshell> /exit| Goodbye
```

TOPS Technologies

JAVA

3) Creating and using functions in Jshell

-Create a function SquareNo() to take int and return its square value.

```
jshell> int SquareNo(int i)
      {
         return i*i;
      }
|  created method SquareNo (int)
jshell> SquareNo (3)
$3 ==> 9
jshell>
```

TOPS Technologies

JAVA

4) Creating variables in Jshell

```
jshell> int i=10;
 i==> 10
jshell>i
 i ==> 10
jshell> int j=300;
j==> 300
jshell>j
j ==> 300
jshell>
```

5) To list all variables and methods

```
jshell> /vars
|int i=10
|int j=300

jshell>/methods
|int SquareNo (int)
```

TOPS Technologies

JAVA

Java 9 Factory Methods

- Static factory methods for List, Set and Map interface.
- These methods are useful to create small number of collection.
- It means we can use these methods to create list, set and map of small number of elements.
- It is modifiable, so adding new element will throw **java.lang.UnsupportedOperationException**

TOPS Technologies

JAVA

Java List interface-

- It is an ordered collection of objects in which duplicate values can be stored.
- Since List preserves the insertion order, it allows positional access and insertion of elements.

Java Set interface-

- It is immutable
- No null elements
- It is serializable if all elements are serializable.
- No duplicate elements.
- The iteration order of set elements is unspecified and is subject to change.

Java Map interface-

- It is immutable
- It does not allow null keys and values
- It is serializable if all keys and values are serializable
- It rejects duplicate keys at creation time
- The iteration order of mappings is unspecified and is subject to change.

JAVA

In Java 9

List.of() factory method for List Interface

Set.of() factory method for Set Interface

Map.of() factory method for Map Interface

TOPS Technologies

JAVA

Java 9 Anonymous Inner Classes

- It is an inner class without a name and for which only a single object is created.

- Anonymous inner class are mainly created in two ways:
  - Class (may be abstract or concrete)
  - Interface

TOPS Technologies

JAVA

Java 10 Updates

- New API added for Collection (copyOf() method)
- Local variable type interface and its rules.

TOPS Technologies

JAVA

Java 10-copyOf() method

- List, Map & Set Interfaces are added with a static copyOf(Collection) method.
- Its returns an modifiable List, Map or Set containing the entries provided.
- For a List, if the given List is subsequently modified, the returned List will not reflect such modifications

JAVA

Local Variable Type Inference

- **Type inference** refers to the automatic detection of the data type of a variable, done generally at the compiler time.
- **Local Variable Type Interface** Allows the developer to skip the **type** declaration associated with **local variables** (those defined inside method definitions, initialization blocks, for-loops, and other blocks like if-else).
- It is job of the compiler to figure out the data type of the variable.

JAVA

In Java 9, to define a local variables of class type, the following was the only correct syntax:

Class_name variable_name=new Class_name(arguments);

For example:

```
class A {

    public static void main(String a[])
    {
        String s = " Hi there";
       System.out.println("String is:"+s);
    }
```

JAVA

In Java 10-
Can be re-written as:

```
// Declaration of a local variable in java 10 using LVTI
class A {
    public static void main(String a[])
    {
        var x = "Hi there";
        System.out.println(x);
    }
}
```

TOPS Technologies

JAVA

Rules for writing Local variable Interference-

1) Not Permitted in class field.

e.g-

```
 class A {
var x; /* Error: class variables can't be declared
        using 'var'. Datatype needs
         to be explicitly mentioned*/
}
```

2) Not permitted for uninitialized local variable.

e.g-

```
class A {
   public static void main(String a[])
   {
      var x; /* error: cannot use 'var'
             on variable without initializer*/
   }
}
```

TOPS Technologies

JAVA

Rules for writing Local variable Interference-

3) Not permitted in method return type.

e.g-

```
class A {
   public var show() /* Error: Method return type
                              can't be var*/
   {
      return 1;
   }
}
```

4) Not permitted with variable initialized with NULL.

e.g-

```
class A {
   public static void main(String a[])
   {
      var x = NULL; // Error: variable initializer is 'null'
   }
```

Java GUI

- Awt
- Swing
- Componets,Containers,Frame,Window,Panel
- Layouts
- Events
- Event handling

TOPS Technologies

AWT

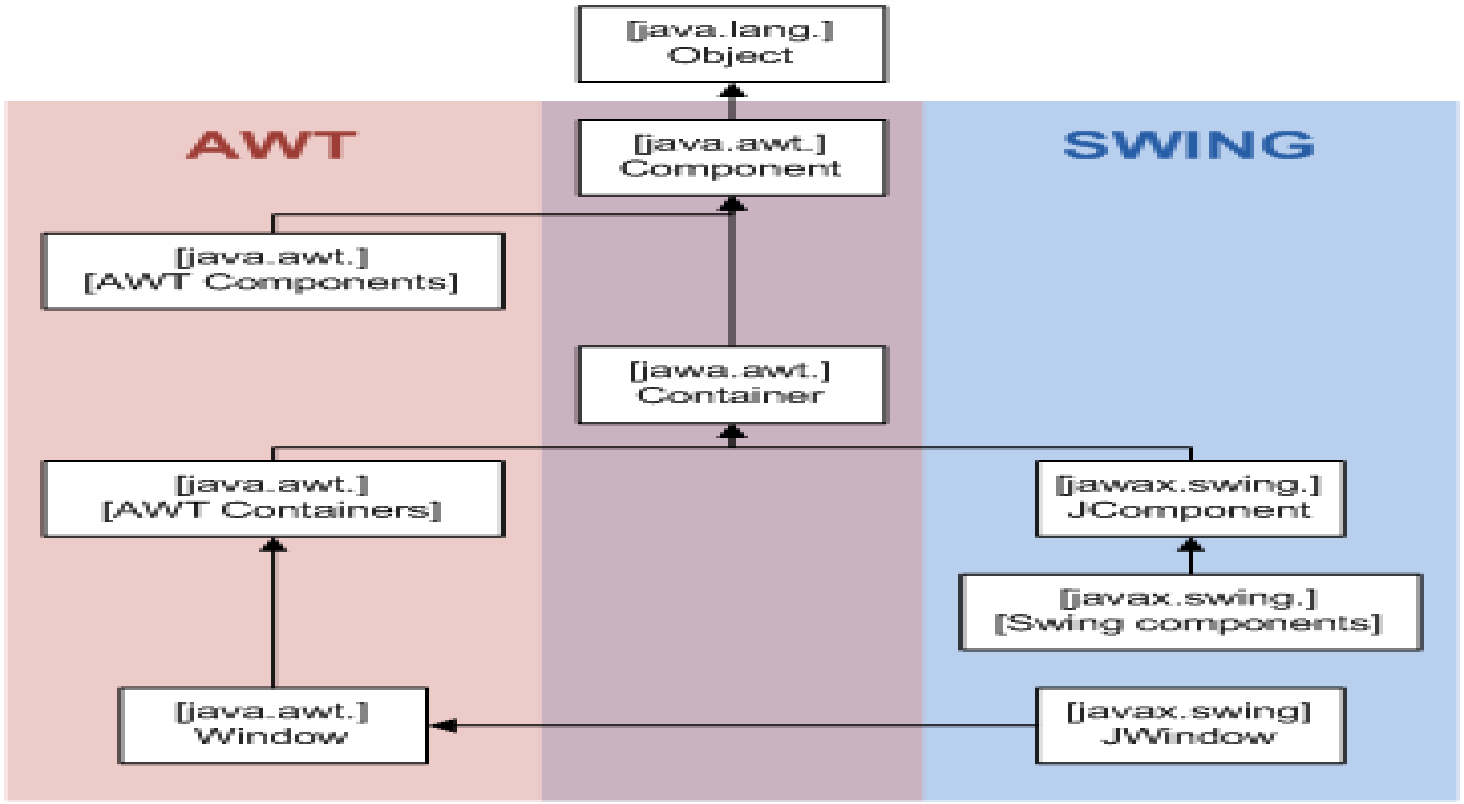
- The **Abstract Window Toolkit** provides many classes for programmers to use.
- The AWT was designed to provide a common set of tools for graphical user interface design that work on a variety of platforms.
- The user interface elements provided by the AWT are implemented using each platform's native GUI toolkit, thereby preserving the look and feel of each platform.
- This is one of the AWT's strongest points.
- The disadvantage of such an approach is the fact that a graphical user interface designed on one platform may look different when displayed on another platform.

Swing

- Swing is a principal GUI toolkit for the Java programming language.
- It is a part of the JFC (Java Foundation Classes), which is an API for providing a graphical user interface for Java programs.
- It is completely written in Java.

TOPS Technologies

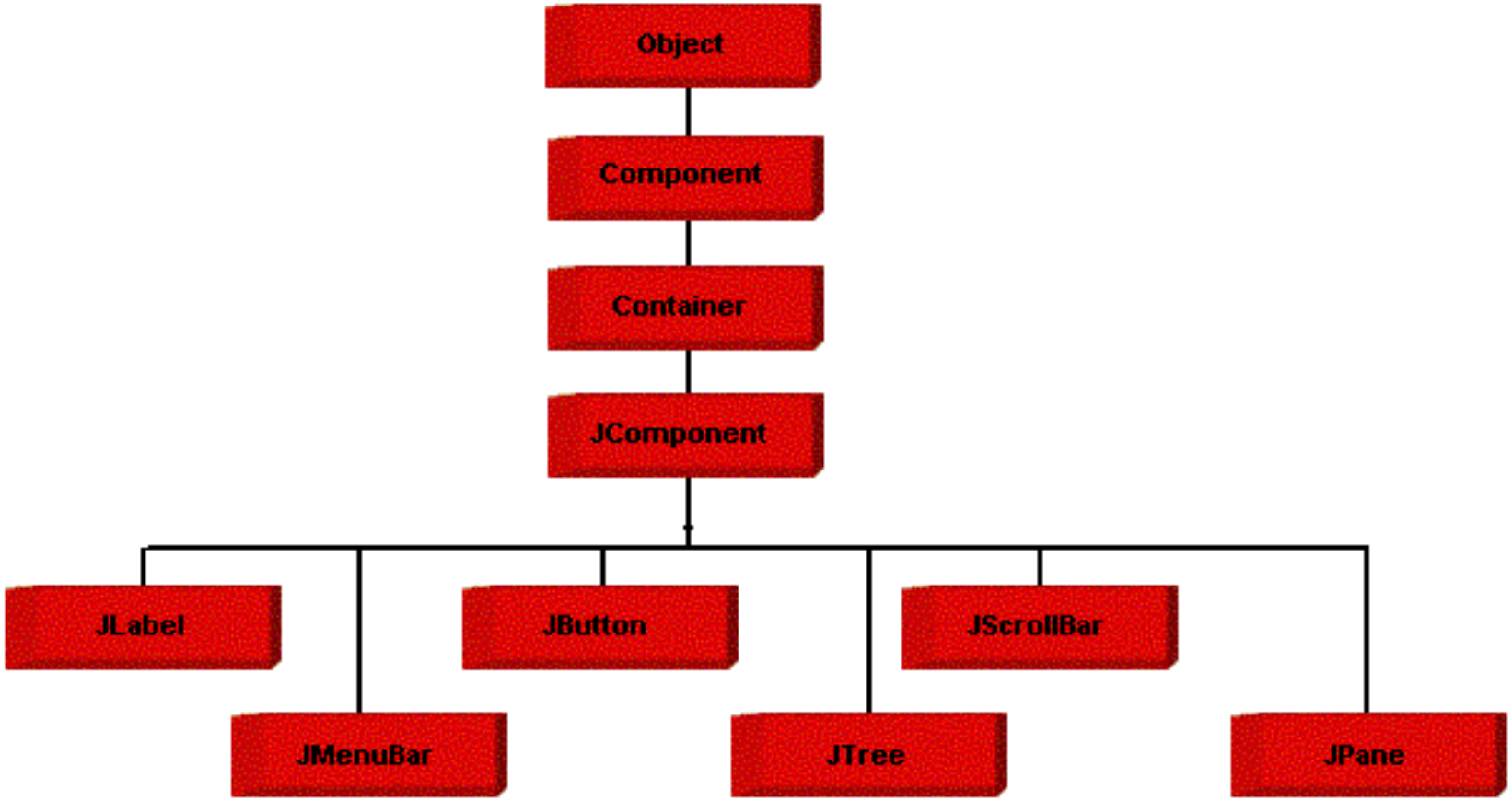
Swing


[java.lang.]
Object
AWT
[java.awt.]
Component
SWING
[java.awt.]
[AWT Components]
[jawa.awt.]
Container
[java.awt.]
[AWT Containers]
[jawax.swing.]
JComponent
[javax.swing.]
[Swing components]
[java.awt.]
Window
[javax.swing]
JWindow

AWT vs Swing

AWT	Swing
AWT components are platform dependent.	Swing components are made in purely java and they are platform independent.
We can not have different look and feel in AWT.	We can have different look and feel in Swing.
AWT components are called Heavyweight component.	Swings are called light weight component because swing components sits on the top of AWT components and do the work.
AWT has limited features compare to swing.	Swing has many advanced features like JTabel, Jtabbed pane which is not available in AWT

Componets,Containers,Frame,Window,Panel



- Swing components are basic building blocks of an application.
- Swing toolkit has a wide range of various components.
- Swing component classes are prefixed with the letter J to distinguish them from corresponding AWT classes:
  - JButton in Swing
  - Button in AWT

- The GUI classes can be classified into three groups:
  - Container classes
    - An abstract class that extends Component.
    - Containers can hold multiple components.
    - Eg: JFrame, JDialog, JApplet, JPanel
  - Helper classes
    - Are not subclasses of Component:
    - Used to describe the properties of GUI components:
    - Eg: Graphics,Color,Font,FontMetrics,Dimension,LayoutManager
  - Component classes
    - An abstract class for GUI components such as JMenu, JButton, JLabel etc.

JFrame

- In the Java Swing, top-level windows are represented by the **JFrame** class.
- Java supports the look and feel and decoration for the frame.
- It can be moved, resized, iconified.
- It is not a subclass of JComponent.
- **Example :**
- https://github.com/TopsCode/Java/blob/master/Module-2/2.14%20Swing/2.14.1%20MyFrame.java

TOPS Technologies

Layouts

- Layouts tell Java where to put components in containers (JPanel, content pane, etc).
- Every panel (and other container) has a default layout, but it's better to set the layout explicitly for clarity.
- To create layouts, we use layout managers.
- Layout managers are one of the most difficult parts of modern GUI programming.
- Several AWT and Swing classes provide layout managers for general use:
- BorderLayout
  - BoxLayout
  - CardLayout
  - FlowLayout
  - GridBagLayout
  - GridLayout

BorderLayout

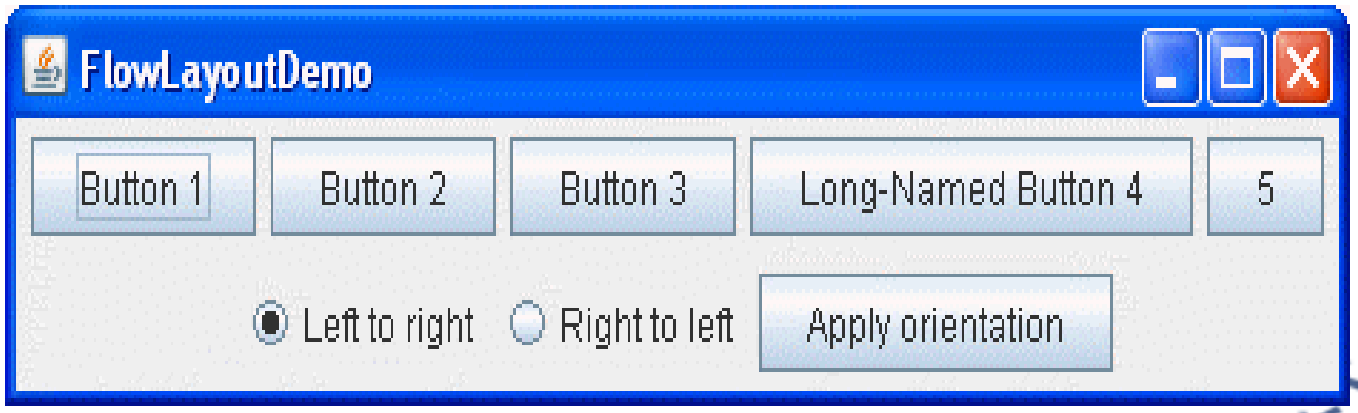
Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.14%20Swing/2.14.2%20BorderLayout.java

FlowLayout

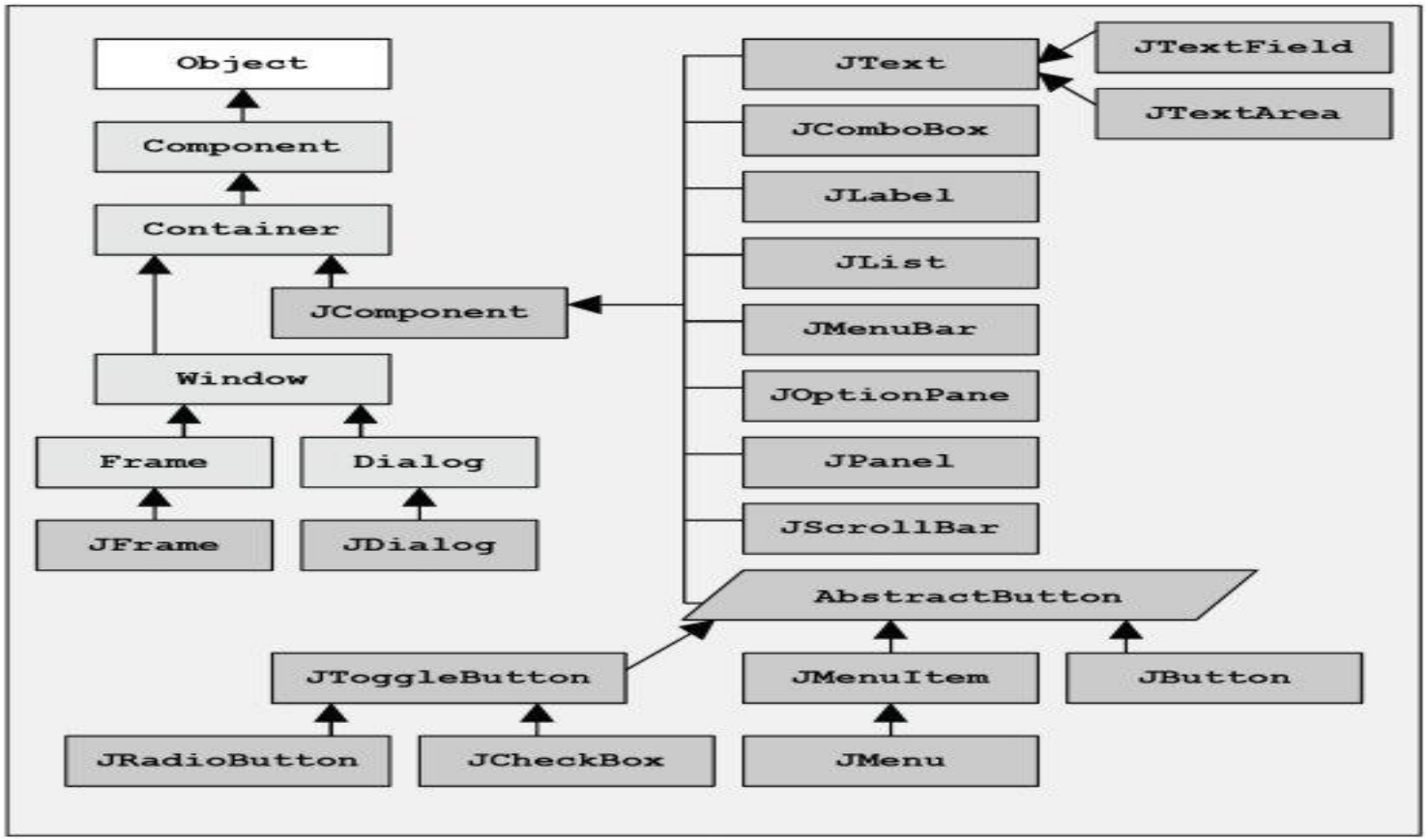
Example :

https://github.com/TopsCode/Java/blob/master/Module-2/2.14%20Swing/2.14.3%20FlowLayout.java



TOPS Technologies

Components


Object
Component
Container
JComponent
Window
Frame
Dialog
JFrame
JDialog
JText
JTextField
JTextArea
JComboBox
JLabel
JList
JMenuBar
JOptionPane
JPanel
JScrollBar
AbstractButton
JToggleButton
JMenuItem
JButton
JRadioButton
JCheckBox
JMenu

JLabel

- A JLabel object provides text instructions or information on a GUI — display a single line of read-only text, an image or both text and image.

Constructor	Description
JLabel()	Creates a JLabel instance with no image and with an empty string for the title.
JLabel(Icon image)	Creates a JLabel instance with the specified image.
JLabel(Icon image, int horizontalAlignment)	Creates a JLabel instance with the specified image and horizontal alignment.
JLabel(String text)	Creates a JLabel instance with the specified text.

JButton

- A button is a component the user clicks to trigger a specific action.

Constructor	Description
JButton()	Creates a button with no set text or icon.
JButton(Action a)	Creates a button where properties are taken from the Action supplied.
JButton(Icon icon)	Creates a button with an icon.
JButton(String text)	Creates a button with text.
JButton(String text, Icon icon)	Creates a button with initial text and an icon.

JTextField

- JTextField allows editing/displaying of a single line of text.

JTextField()	Constructs a new TextField.
JTextField(Document doc, String text, int columns)	Constructs a new JTextField that uses the given text storage model and the given number of columns.
JTextField(int columns**)**	Constructs a new empty TextField with the specified number of columns.
JTextField(String text)	Constructs a new TextField initialized with the specified text.
JTextField(String text, int columns)	Constructs a new TextField initialized with the specified text and columns.

```
JTextField txtName=new JTextField(15);
```

JTextArea

- JTextArea allows editing of multiple lines of text.

JTextArea()	Constructs a new TextArea.
JTextArea(Document doc)	Constructs a new JTextArea with the given document model, and defaults for all of the other arguments
JTextArea(Document doc, String text, int rows, int columns)	Constructs a new JTextArea with the specified number of rows and columns, and the given model.
JTextArea(int rows, int columns)	Constructs a new empty TextArea with the specified number of rows and columns.

JPasswordField

- JPasswordField (a direct subclass of JTextField) you can suppress the display of input.

JPasswordField()	Constructs a new JPasswordField, with a default document, null starting text string, and 0 column width.
JPasswordField(Document doc, String txt, int columns)	Constructs a new JPasswordField that uses the given text storage model and the given number of columns.
JPasswordField(int columns)	Constructs a new empty JPasswordField with the specified number of columns.
JPasswordField(String text)	Constructs a new JPasswordField initialized with the specified text.
JPasswordField(String text, int columns)	Constructs a new JPasswordField initialized with the specified text and columns.

JRadioButton

- A set of radio buttons can be associated as a group in which only one button at a time can be selected.

JRadioButton()	Creates an initially unselected radio button with no set text.
JRadioButton(Action a)	Creates a radiobutton where properties are taken from the Action supplied.
JRadioButton(Icon icon)	Creates an initially unselected radio button with the specified image but no text.
JRadioButton(Icon icon, boolean selected)	Creates a radio button with the specified image and selection state, but no text.
JRadioButton(String text)	Creates an unselected radio button with the specified text.

JCheckBox

- Acheckbox can be selected and deselected, and it also displays its current state.

Constructor	Discription
JCheckBox()	Creates an initially unselected check box button with no text, no icon.
JCheckBox(Action a)	Creates a check box where properties are taken from the Action supplied.
JCheckBox(Icon icon)	Creates an initially unselected check box with an icon.
JCheckBox(Icon icon, boolean selected)	Creates a check box with an icon and specifies whether or not it is initially selected.
JCheckBox(String text)	Creates an initially unselected check box with text.
JCheckBox(String text, boolean selected)	Creates a check box with text and specifies whether or not it is initially selected.

JComboBox

- JComboBox is like a drop down box — you can click a drop-down arrow and select an option from a list.

JComboBox()	Creates a JComboBox with a default data model.
JComboBox(ComboBoxModel aModel)	Creates a JComboBox that takes it's items from an existing ComboBoxModel.
JComboBox(Object[] items)	Creates a JComboBox that contains the elements in the specified array.
JComboBox(Vector items)	Creates a JComboBox that contains the elements in the specified Vector.

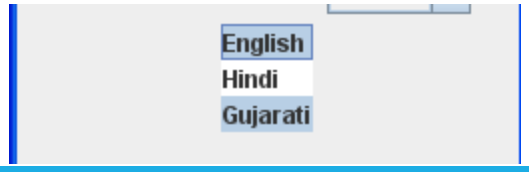
```
String languages[]={"English","Hindi","Gujarati"};
JComboBox language=new JComboBox(languages);
```

JList

- JList provides a scrollable set of items from which one or more may be selected.

JList()	Constructs a JList with an empty model.
JList(ListModel dataModel)	Constructs a JList that displays the elements in the specified, non-null model.
JList(Object[] listData)	Constructs a JList that displays the elements in the specified array.
JList(Vector listData)	Constructs a JList that displays the elements in the specified Vector.

JList langList=new JList(languages);



JScrollPane

- Provides a scrollable view of a lightweight component.

JScrollPane()	Creates an JScrollPane where both horizontal and vertical scrollbars appear when needed.
JScrollPane(Component view)	Creates a JScrollPane that displays the contents of the specified component, where both horizontal and vertical scrollbars appear whenever the component's contents are larger than the view.
JScrollPane(Component view, int vsbPolicy, int hsbPolicy)	Creates a JScrollPane that displays the view component in a viewport whose view position can be controlled with a pair of scrollbars.
JScrollPane(int vsbPolicy, int hsbPolicy)	Creates an empty (no viewport view) JScrollPane with specified scrollbar policies.

JDialog

- The main class for creating a dialog window. You can use this class to create a custom dialog,
- Dialogs are important means of communication between a user and a computer program.

JDialog()	Creates a modeless dialog without a title and without a specified Frame owner.
JDialog(Dialog owner)	Creates a modeless dialog with the specified Dialog as its owner and an empty title.
JDialog(Frame owner, String title)	Creates a modeless dialog with the specified title and with the specified owner frame.

JMenuBar

- An implementation of a menu bar.
- You add JMenu objects to the menu bar to construct a menu.
- JMenuBar menuBar = new JMenuBar();
- frame.setJMenuBar(menuBar);

Constructor	Discription
JMenuBar()	Creates a new menu bar.

JMenu

- An implementation of a menu -- a popup window containing JMenuItems that is displayed when the user selects an item on the JMenuBar.

Constructor	Description
JMenu()	Constructs a new JMenu with no text.
JMenu(Action a)	Constructs a menu whose properties are taken from the Action supplied.
JMenu(String s)	Constructs a new JMenu with the supplied string as its text.
JMenu(String s, boolean b)	Constructs a new JMenu with the supplied string as its text and specified as a tear-off menu or not.

JMenuItem

- An implementation of an item in a menu.
- A menu item is essentially a button sitting in a list.

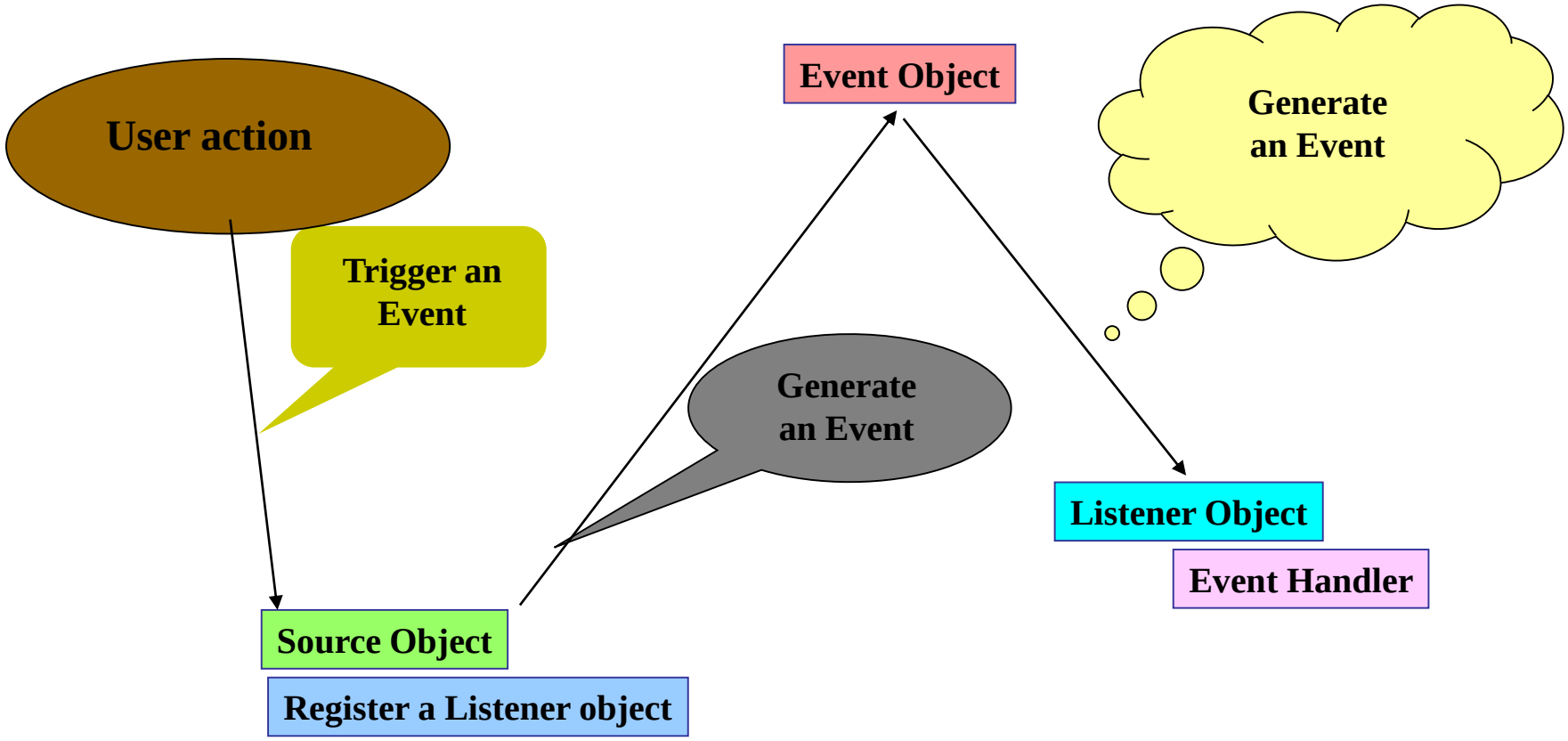
Constructor	Description
JMenuItem()	Creates a JMenuItem with no set text or icon.
JMenuItem(Action a)	Creates a menu item whose properties are taken from the specified Action.
JMenuItem(Icon icon)	Creates a JMenuItem with the specified icon.
JMenuItem(String text)	Creates a JMenuItem with the specified text.
JMenuItem(String text, Icon icon)	Creates a JMenuItem with the specified text and icon.
JMenuItem(String text, int mnemonic)	Creates a JMenuItem with the specified text and keyboard mnemonic.

Example of All Components :

https://github.com/TopsCode/Java/blob/master/Module-2/2.14%20Swing/2.14.5%20RegistrationForm.java

Event Handling

- Any program that uses GUI (graphical user interface) such as Java application written for windows, is event driven.
- Event describes the change of state of any object.
- **Example :** Pressing a button, Entering a character in Textbox.
- **Event Components**
- **Events :** An event is a change of state of an object.
- **Events Source :** Event source is an object that generates an event.
- **Listeners :** A listener is an object that listens to the event. A listener gets notified when an event occurs



An event triggered by user actions on the source objects. The source object generates the event object and invokes the handler of the listener object to process the event.

Event Classes	Description	Listener Interface	Event Handler
ActionEvent	generated when button is pressed, menu-item is selected, list-item is double clicked	ActionListener	Action Performed
KeyEvent	generated when input is received from keyboard	KeyListener	Key Pressed (Key event e) Key released (Key Event e) Key typed (Key Event e)
ItemEvent	generated when check-box or list item is clicked	ItemListener	Item Statechanged (item Event e)
TextEvent	generated when value of textarea or textfield is changed	TextListener	Text Value Changed (Text Event e)

EventHandling Codes

- We can put the event handling code into one of the following places:
  - By implementing interface
  - By extending adaptor class
  - By anonymous adaptor class
  - By anonymous in interface .

By extending adaptor class

- Every listener that includes more than one abstract method has got a corresponding adapter class.
- The advantage of adapter is that we can override any one or two methods we like instead of all.
- But incase of a listener, we must override all the abstract methods
- **Example :**

- https://github.com/TopsCode/Java/blob/master/Module-2/2.14%20Swing/2.14.6%20EventHandlingWithClass.java

Listener Interface	Corresponding Adapter
WindowListener	WindowAdapter
MouseListener	MouseAdapter
MouseMotionListener	MouseMotionAdapter
KeyListener	KeyAdapter
FocusListener	FocusAdapter

ActionListener, **ItemListener** and **TextListener** do not have a corresponding adapter class as they contain only one abstract method.
The disadvantage of the adaptor class is that if we want extend another class then java doesn't provide it.(i.e multiple inheritance is not allowed).

Example

https://github.com/topscode/java/blob/master/module-2/2.14%20swing/2.14.7%20eventhandlingwithinterface.java

https://github.com/topscode/java/blob/master/module-2/2.14%20swing/2.14.8%20eventwithanonymusclass.java

To See the example

https://github.com/topscode/java/blob/master/module-2/2.14%20swing/2.14.9%20eventwithanonymusinterface.java

Module 3 [RDBMS And Database programming using JDBC]

- DBMS &RDBMS
- MYSQL
- Introduction to JDBC
- Result Set Interface
- Database Metadata
- Result Set MetaData

TOPS Technologies

RDBMS

- RDBMS – relational database management system
- In RDBMS, all data are stored in tables and relationship between data tables are stored in another tables.
- RDBMS was provided by Dr.E.F.Codd, he has given 12 rules for standard RDBMS. (which has not been met in any database system yet!)

TOPS Technologies

Queries Type

- All SQL Commands are divided into four part based on functionality
    - DDL
    - DML
    - DQL
    - DCL

DDL Statement

- DDL
  - DDL is data definition language.
  - It is used to define structure of database and tables.
  - We can create, modify or delete structure of tables.\

DDL Statement	Description	syntax	example
Create	Create command is used to create structure of table.	create table <table_name> (<columnName1> <datatype>(<size>));	create table Person_Master(Name nvarchar(50),Age numeric(18,0));
Alter	Once you have defined structure, to modify that Alter command is used.	alter table <tableName> **add**(<NewColumnName> <dataType>(<Size>),<NewColumn Name><datatype>(<size>)..);	alter table Person_Master add Address nvarchar(50);

Command	Description	Syntax	example
alter			alter table Person_Master drop column Age;
		alter table <tableName> **alter** (<columnName> <DataType> <newSize>));	alter table Person_Master alter column Name varchar(30);
Truncate	Truncate will empty table completely. It will drop table first and then will recreate structure of table.	truncate table <tableName>;	truncate table Person_Master;
Drop	This command will discard whole table.	drop table <tableName>;	drop table Person_Master;

DML Statement

- DML is data manipulation language. It is used to manipulate data inside table.

Command	Description	Syntax	example
Insert	This command is used to insert records inside table.	insert into <tableName>(columnName1, columnName2,..columnNamen)values(value1,value2,..valuen);	insert into Person_Master(Name,Age, Address)values('Name1',21 ,'Address1');
Update	This command is used to modify data inside table.	update <tableName> set <columnName1>=value1,<columnName2>=value2..<columnNamen=valuen;	update Person_Master set Name='name11';
Delete	This command is used to delete records from table.	delete from <tableName>;	delete from Person_Master;

DQL Statement

- DQL stands for **Data Query Language.**
- DQL Command is : Select

Command	Description	syntax	example
Select	This will select ‘n’ columns from table.	select <columnName1>,<columnName2>,..<ColumnNamen> from <tableName>;	select Name,Age,Address from Person_Master;
	To select all records from database,	select * from <tableName>;	select * from Person_Master;

DCL Statement

- Data Control Language

command	description	syntax	Example	Example description
Commit	Commit is saving of result of all actions or queries.	commit;	START TRANSACTION ; delete from Student_Master where id =102 Commit	Than after execute rollback command it will not show deleted record;
Rollback	Once any transaction has been done, to undo that transaction, rollback can be used.	rollback;	START TRANSACTION delete from Student_Master where id =102 rollback	Deleted record will be roll backed(i.e undo)

Views

- Once table is created and data is populated we may need to hide some data from some users.
- This leads to data security concept.
- For this, we can create views.
- View is sub set of database table, it means it is type of table but containing required part of data only.It is virtual table.

Syntax

- create view <viewName> as select <columnName1>,<columnName2> from <tableName>;\

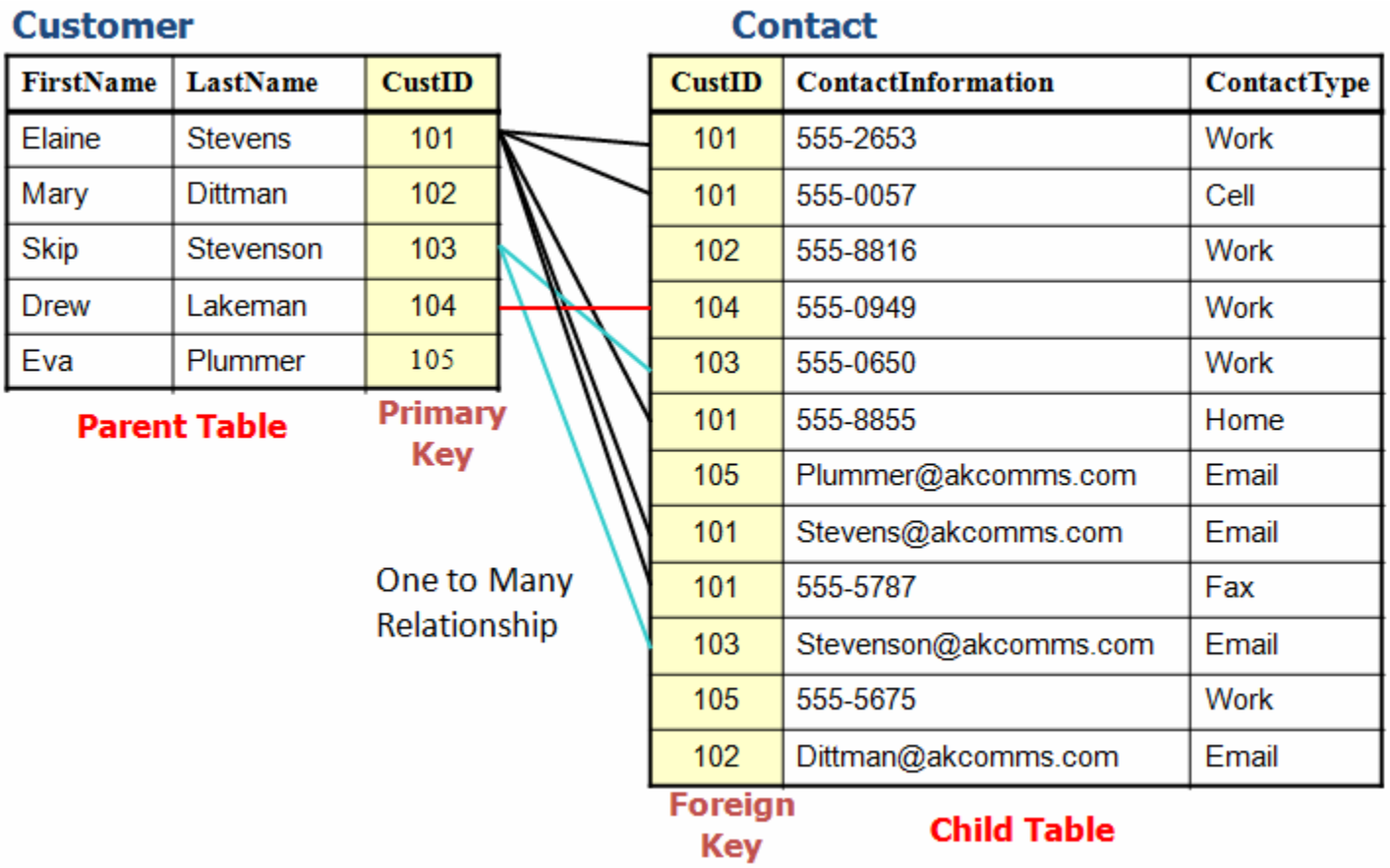
Example of Quires

https://github.com/TopsCode/Java/blob/master/Module-3/3.1BasicQueries.txt

TOPS Technologies

Constraint

- To avoid duplication, we need to define constraint on data.
- For example, all students will have unique roll number.
- Like
  - primary key,
  - foreign key,
  - unique key etc.


Customer
FirstName
LastName
CustID
Elaine
Stevens
101
Mary
Dittman
102
Skip
Stevenson
103
Drew
Lakeman
104
Eva
Plummer
105
Parent Table
Primary Key
One to Many Relationship
Contact
CustID
ContactInformation
ContactType
101
555-2653
Work
101
555-0057
Cell
102
555-8816
Work
104
555-0949
Work
103
555-0650
Work
101
555-8855
Home
105
Plummer@akcomms.com
Email
101
Stevens@akcomms.com
Email
101
555-5787
Fax
103
Stevenson@akcomms.com
Email
105
555-5675
Work
102
Dittman@akcomms.com
Email
Foreign Key
Child Table

- Primary keys and foreign keys are two types of constraints that can be used to enforce data integrity in SQL Server tables. These are important database objects.
- Primary Key can be defined while creating a table with Create Table command or it can be added with the Alter table command.

Primary key	Foreign Key
Primary key uniquely identify a record in the table.	Foreign key is a field in the table that is primary key in another table.
Primary Key can't accept null values.	Foreign key can accept multiple null value.
We can have only one Primary key in a table.	We can have more than one foreign key in a table.
CREATE TABLE Department (DeptID int PRIMARY KEY, Name varchar (50) NOT NULL, Address varchar(100) NULL)	CREATE TABLE Employee (EmpID int PRIMARY KEY, Name varchar (50) NOT NULL, Salary int NULL, DeptID int FOREIGN KEY REFERENCES Department(DeptID))

Primary Key	Unique Key
A primary key *cannot allow* null values. (You cannot define a primary key on columns that allow nulls.)	A unique key *can allow* null values. (You can define a unique key on columns that allow nulls.)
Each table can have *at most one* primary key.	Each table can have *multiple* unique keys.
On some RDBMS a primary key automatically generates a *clustered* table index by default.	On some RDBMS a unique key automatically generates a *non-clustered* table index by default.
CREATE TABLE Persons (P_Id int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255), City varchar(255), PRIMARY KEY (P_Id))	CREATE TABLE Person_Master (P_Id int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255), City varchar(255), CONSTRAINT uc_PersonID UNIQUE (P_Id,LastName))

Joins

- Join, Inner Join, Cross Join, Outer Join
- OUTER Join further divides into Left Join, Right Join and Full Join.

Example of Joins

https://github.com/TopsCode/Java/blob/master/Module-3/3.2Joins.txt

Procedure & function

- They are logically grouped SQL or Pl/SQL commands that performs some task.
- Creating stored procedure:

Syntext

create procedure [schema].<procedureName> (<argument> In,out,In,out}<datatype>) {Is As}

<variable>declaration;

Begin <Queries>

Exception <exception part>

End;

Example

https://github.com/TopsCode/Java/blob/master/Module-3

TOPS Technologies

Creating functions :

create function [schema].<functionName>(<argument> in <datatype>..)

return <datatype> {is as} <variable>declaration;

Begin <SQL queries>;

Exception <exception part>;

End;

Example

https://github.com/TopsCode/Java/blob/master/Module-3/3.4FUNCTION.txt

TOPS Technologies

Trigger

- A SQL trigger is a set of SQL statements stored in the database catalog.
- A SQL trigger is executed or fired whenever an event that is associated with a table occurs e.g., insert, update or delete.
- A SQL trigger is a special type of stored procedure.
- It is special because it is not called directly like a stored procedure.
- The main difference between a trigger and a stored procedure is that a trigger is called automatically when a data modification event is made against a table whereas a stored procedure must be called explicitly.
- Example :
- https://github.com/TopsCode/Java/blob/master/Module-3/3.5TRIGGER.txt

- MySQL allows you to define maximum six triggers for each table.
- **BEFORE INSERT** – activated before data is inserted into the table.
- **AFTER INSERT-** activated after data is inserted into the table.
- **BEFORE UPDATE** – activated before data in the table is updated.
- **AFTER UPDATE -** activated after data in the table is updated.
- **BEFORE DELETE** – activated before data is removed from the table.
- **AFTER DELETE** – activated after data is removed from the table.

TOPS Technologies

Java Database Programming

- JDBC Introducation
- JDBC Architecture
- Driver Types
- Steps for Creating Connections
- Types of Statements
- ResultSet Interface
- DatabaseMetaData
- ResultSetMetadata
- Example of All database opration with swing
- Example of DatabaseMetaData

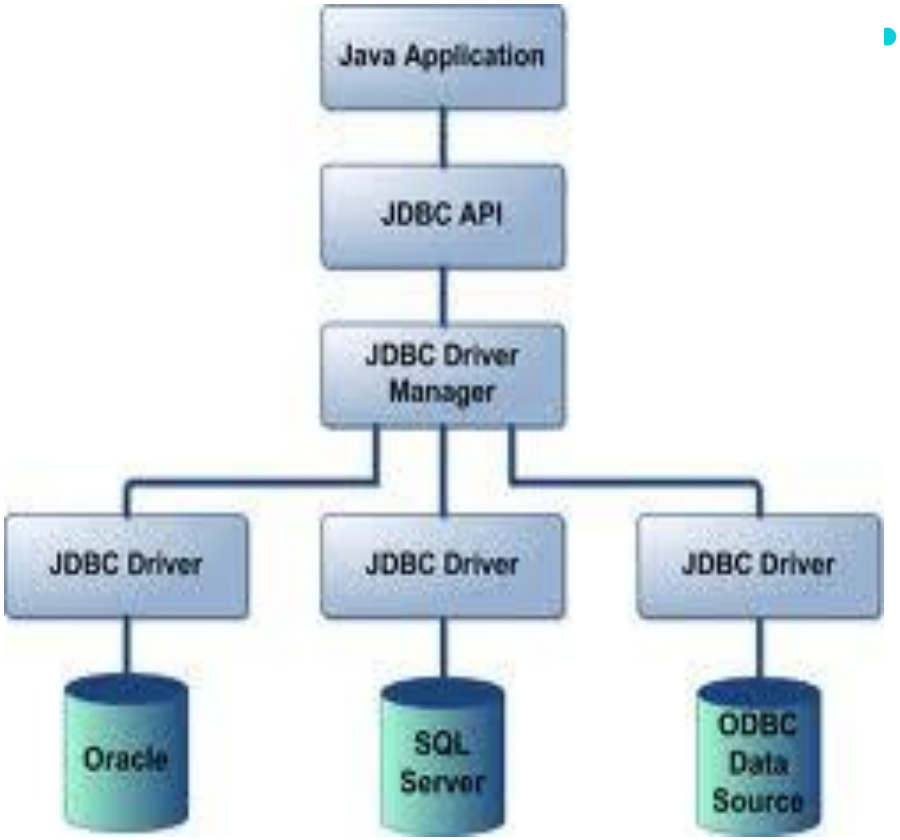
What is JDBC??

- JDBC is Java application programming interface that allows the Java programmers to access database management system from Java code. It was developed by JavaSoft, a subsidiary of Sun Microsystems.

- JDBC provides a standard library for accessing relational databases

TOPS Technologies

JDBC Architecture



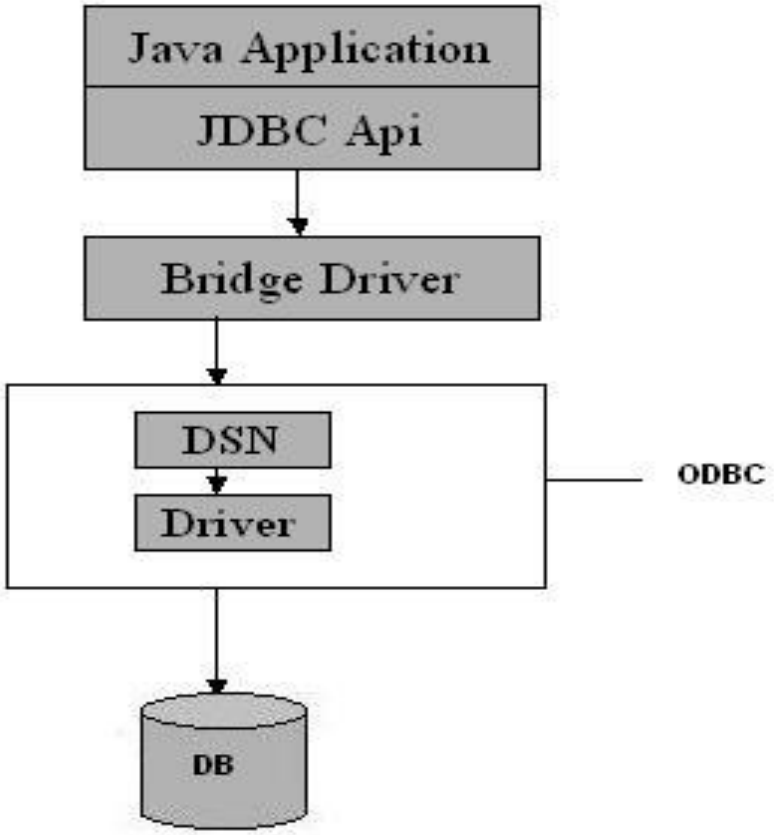
- JDBC Consist Two Main Part:
  - JDBC API – a purely java based api
  - JDBC Driver Manger - communicate with vendor-specific drivers that perform the real communication with database

JDBC Driver Types

- Driver types are used to categorize the technology used to connect to the database
- There are four types of JDBC drivers known as:
  - Type 1 (JDBC-ODBC bridge driver)
  - Type 2 (Native-API, partly Java driver)
  - Type 3 (JDBC-Net, pure Java driver)
  - Type 4 (Native-protocol, pure Java driver)

TOPS Technologies

Type 1 Driver

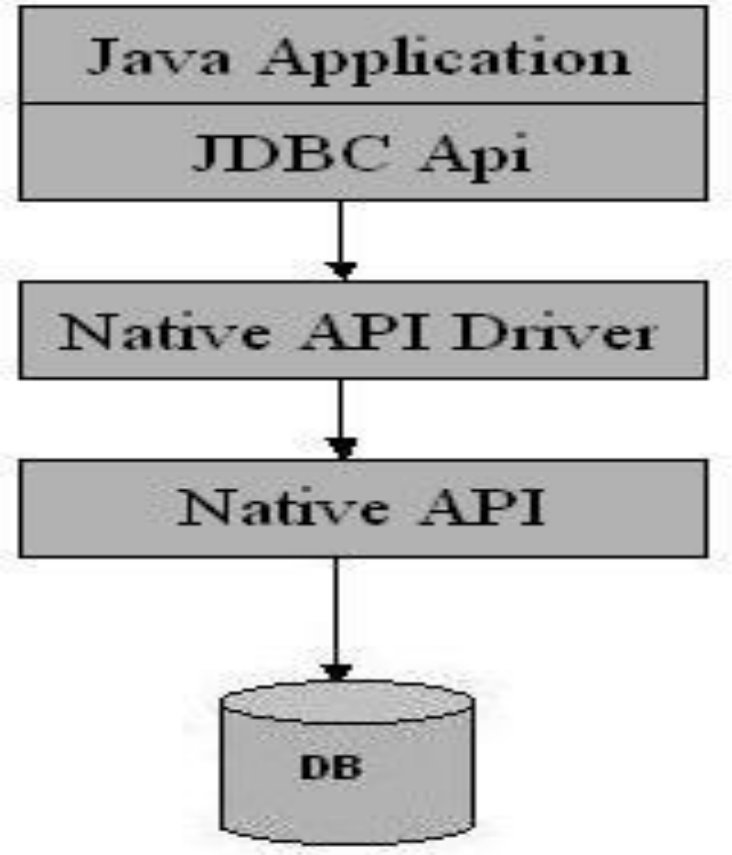


- known as JDBC-ODBC Bridge
- Type 1 driver translates all JDBC calls into ODBC calls and sends them to the ODBC driver

Disadvantage:

- Since the Bridge driver is not written fully in Java, Type 1 drivers are not portable

Type 2 Driver

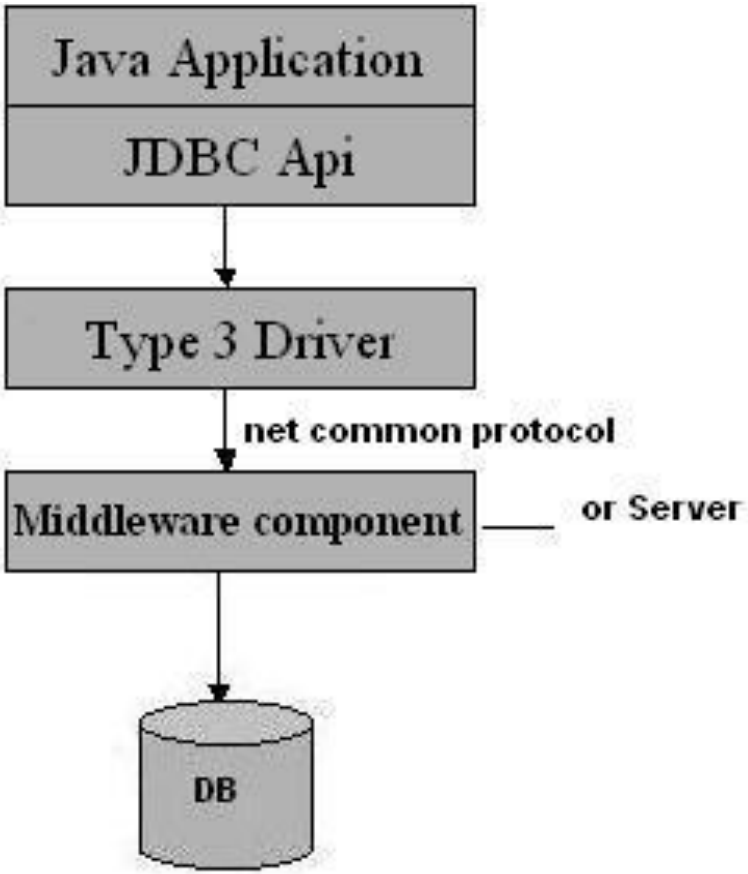


- convert JDBC calls into database-specific calls
  - i.e. specific to a particular database

Disadvantage:

- Like Type 1 drivers, it's not written in Java Language which forms a portability issue
- Mostly obsolete now
- Not thread safe

Type 3 Driver

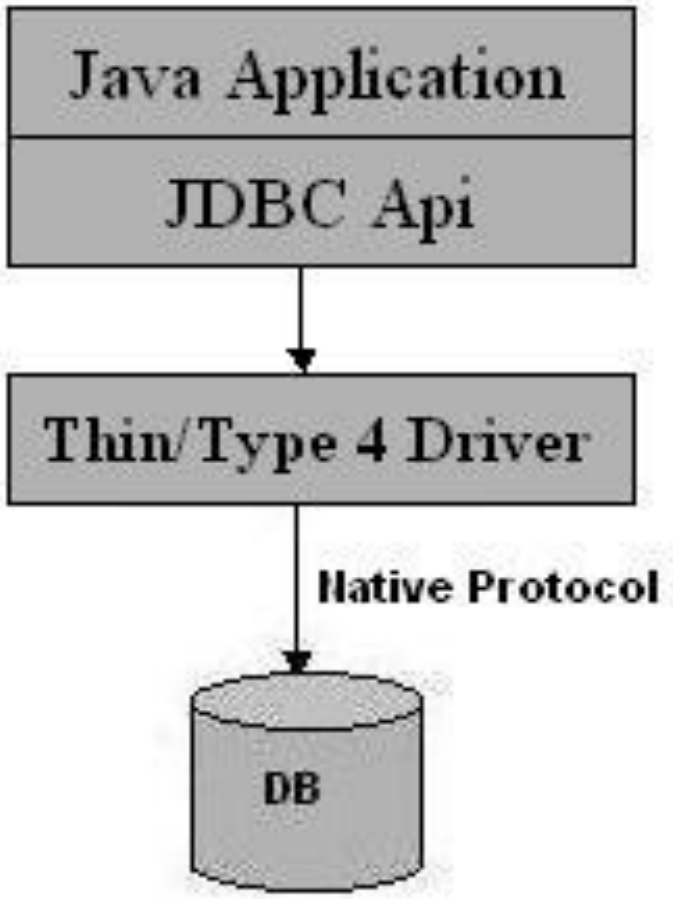


- database requests are passed through the network to the middle-tier server
- The middle-tier then translates the request to the database
- This driver is fully written in Java and hence Portable

<u>Disadvantage:</u>

- It requires another server application to install and maintain

Type 4 Driver



- The Type 4 uses java networking libraries to communicate directly with the database server
- completely written in Java to achieve platform independence
- Number of translation layers is very less
- You don't need to install special software on the client or server

Disadvantage:

- With type 4 drivers, the user needs a different driver for each database.

Steps To Connect With Database

- **Step 1 : Load the Driver**
- **Step 2 : Define the Connection URL**
- **Step 3 : Establish the Connection**
- **Step 4 : Create a Statement**
- **Step 5:Execute a Query**
- **Step 6:Process the Result**
- **Step 7 Close the Connection**

TOPS Technologies

Statements in JDBC

- Stements interface provide differants kinds of Methods which is used for the execute querys.
- They are three types of Statements.

Interface	Use
Statement	For execute simple SQL Statements means Static SQL Statements.
Prepared Statement	For execute precompiled SQL Statements Passing IN parameters.
Callable Statement	For execute datatbase SQL Statements. Passing IN and OUT parameters.

Result Set Interface.

- The SQL statements that read data from a database query return the data in a result set. The SELECT statement is the standard way to select rows from a database and view them in a result set. The *java.sql.ResultSet* interface represents the result set of a database query.
- A ResultSet object maintains a cursor that points to the current row in the result set. The term "result set" refers to the row and column data contained in a ResultSet object.

- **Method of Result Set interface.**

Method	Use
next()	Moves the cursor forward one row from its current position.
previous()	Moves the cursor to the previous row in this ResultSet object.
getString(int columnIndex)	Retrieves the value of the designated column in the current row of this ResultSet object as a String in the Java programming language.
getString(String columnlabel)	Retrieves the value of the designated column in the current row of this ResultSet object as a String in the Java programming language.
And Many more Metods……..	

Example of JDBC

https://github.com/TopsCode/Java/blob/master/Module-3/3.6All.java

Database metadata Interface.

- DatabaseMetaData interface provides methods to get meta data of a database such as database product name, database product version, driver name, name of total number of tables, name of total number of views etc.
- Followings are the Method of DatabaseMetaData Interface.

Example of Database Metadata

https://github.com/TopsCode/Java/blob/master/Module-3/3.7DatabaseMetaData.java

- **Method of DatabaseMetaData interface.**

Method	Use
getDriverName()	it returns the name of the JDBC driver. In String
getDriverVersion()	it returns the version number of the JDBC driver. In string
getUserName()	it returns the username of the database. In string.
getDatabaseProductName()	it returns the product version of the database.
And Many more Metods……..	

Resultset Metadata Interface.

- To query an unknown result set for information about the columns that it contains,
- you need to use ResultSetMetaData methods to determine the characteristics of the ResultSets before you can retrieve data from them.
- ResultSetMetaData methods provide the following types of information:

Example of Result Set Metadata

https://github.com/TopsCode/Java/blob/master/Module-3/3.8ResultSetMetaData.java

- **Method of ResultSetMetaData interface.**

Method	Use
getColumnCount()	returns the number of columns in the ResultSet
getTableName()	returns the qualifier for the underlying table of the ResultSet
getSchemaName()	returns the the designated column's table's schema name
getColumnName()	returns the column name.
And Many more….	

Module 4
Web Technologies in Java

Agenda

- **Introduction of Web Designing Technologies (HTML & CSS)**
- **Introduction of Client side Scripting JavaScript**
- **Introduction of Client and Server Architecture**
- **HTTP Overview with Client and Server Request and Response**
- **J2EE Architecture Explanation**
- **Web Component Development in Java**
  - **CGI Programming Process Advantage & Disadvantage**
  - **Servlet Programming Process Advantage & Disadvantage**
  - **JSP Introduction Advantage and Disadvantage**

HTML Lists

- HTML offers authors several mechanisms for specifying lists of information.
- All lists must contain one or more list elements. Lists may contain:
  - Unordered information.
  - Ordered information.
  - Definitions.

TOPS Technologies

List type	Tag	Attribute	value	Description	Example
Order List		reversed		Specifies that the list order should be descending (9,8,7...)	<ol reversed>
		start	*number*	Specifies the start value of an ordered list	<ol start="10">
		type	1 A a I i	Specifies the kind of marker to use in the list	<ol type="A">

List Type	Tag	Attribute	value	description	example
Unordered List		type	disc square circle	Use to Specifies the kind of bullet use in the list.	<ul type="disc|circle|square">

HTML Hyperlink

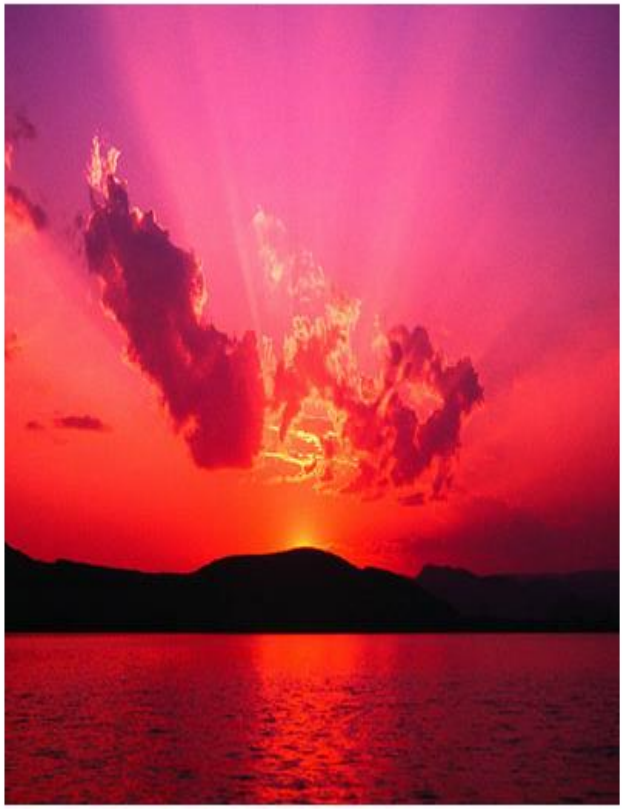
		attribute		example
<a>	This tag defines hyperlink, which is used to link from one page to another.	href	Specifies the URL of the page the link goes to.	Visit Google!
		target values (_blank _parent _self _top)	Specifies where to open the linked document	Visit google!

HTML Image tag

- To add image in html page we can use <img> tag.

attribute	Description	value	example
Align	places the graphic image at a specified position, in relation either to the page margins or to the text.	top bottom middle left right	<img src="sunset.jpg" align="middle">
alt	Specifies an alternate text for an image	*text*	<img src="sunset.jpg" alt="Sun set" >
height	Specifies the height of an image	*pixels*	<img src="sunset.jpg" alt="Smiley face" height="42" width="42">
width	Specifies the width of an image	*pixels*	<img src="sunset.jpg" alt="Smiley face" height="42" width="42">
src	Specifies the URL of an image	URL	<img src="sunset.jpg" >

<img alt="Sunset" width="400" height="400" align="right" src="c:/Documents and Settings/All Users/Documents/My Pictures/Sample Pictures/Sunset.jpg">



HTML table tag

- The HTML <table> element inserts a table in the document.
- This element is the main container of a table, but many other elements are needed to define a table correctly.

attribute	value	example
align	left center right	<table border="1" align="right">
bgcolor	*rgb(x,x,x)* *#xxxxxx* *colorname*	<table border="1" bgcolor="#00FF00">
border	“”	<table border="1" bgcolor="#00FF00">
cellpadding	*pixels*	<table border="1" cellpadding="10">
cellspacing	*pixels*	<table border="1" cellspacing="10">
width	*pixels*, *%*	<table border="1" width="400">

Tag	Attribute	Values	description
<th>	Defines a header cell in a table		
	align	left right center justify char	Aligns the content in a header cell
	bgcolor	*rgb(x,x,x)* *#xxxxxx* *colorname*	Specifies the background color of a header cell
	colspan	*number*	Specifies the number of columns a header cell should span

Tag	Attribute	Value	description
<th>	height	*pixels* *%*	Sets the height of a header cell
	rowspan	*number*	Specifies the number of rows a header cell should span
	width	*pixels* *%*	Specifies the width of a header cell

Table headers:

Name	Telephone
Tops Technologies	123456789

Vertical headers:

First Name:	Tops Technology
Telephone:	123456789

Cell that spans two columns:

Name	Telephone	
Tops Technologies	123456789	987654321

Cell that spans two rows:

First Name:	Tops Technologies
Telephone:	123456789
	987654321

HTML Form tag

- HTML Forms are used to select different kinds of user input.

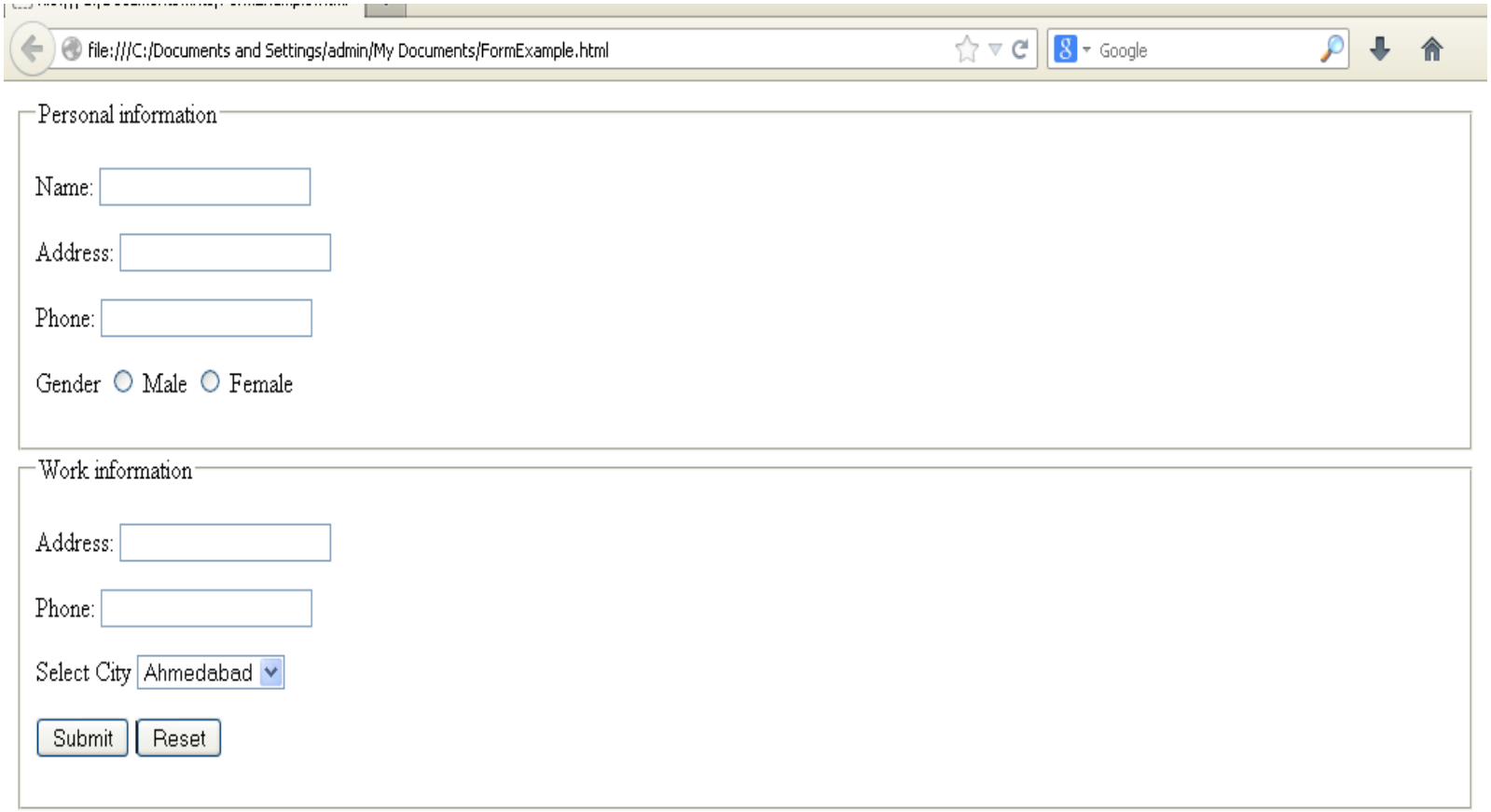
Attribute	Value	Descreption
action	*URL*	Specifies where to send the form-data when a form is submitted
enctype	application/x-www-form-urlencoded multipart/form-data text/plain	Specifies how the form-data should be encoded when submitting it to the server (only for method="post")
method	get post	Specifies the HTTP method to use when sending form-data
name	*text*	Specifies the name of a form
target	_blank _self _parent _top	Specifies where to display the response that is received after submitting the form

Tags	Description
<input>	The <input> tag specifies an input field where the user can enter data.
<textarea>	The <textarea> tag defines a multi-line text input control.
<button>	The <button> tag defines a clickable button.
<select>	The <select> element is used to create a drop-down list.
<option>	The <option> tags inside the <select> element define the available options in the list.
<optgroup>	The <optgroup> is used to group related options in a drop-down list.
<fieldset>	The <fieldset> tag is used to group related elements in a form.
<label>	The <label> tag defines a label for an <input> element.

Tag	Attribute	Description	values
<input>	type	Specifies the type of control.	text password checkbox radio submit reset file hidden image button
	name	Assigns a name to the input control.	
	value	Specifies the intial value for the control. **Note**: If type="checkbox" or type="radio" this attribute is required.	

tag	Attribute	Description
<input>	size	Specifies the width of the control. If type="text" or type="password" this refers to the width in characters. Otherwise it's in pixels.
	maxlength	Specifies the maximum number of characters that the user can input. This can be bigger than the value indicated in the size attribute.
	checked	If type="radio" or type="checkbox" it will already be selected when the page loads.
	src	If type="image", this attribute specifies the location of the image.

HTML Form tag Example



CSS

- CSS Introduction
- Types of CSS
- Pseudo-Classes
- Margins and Padding
- CSS background
- CSS using ID and Class.

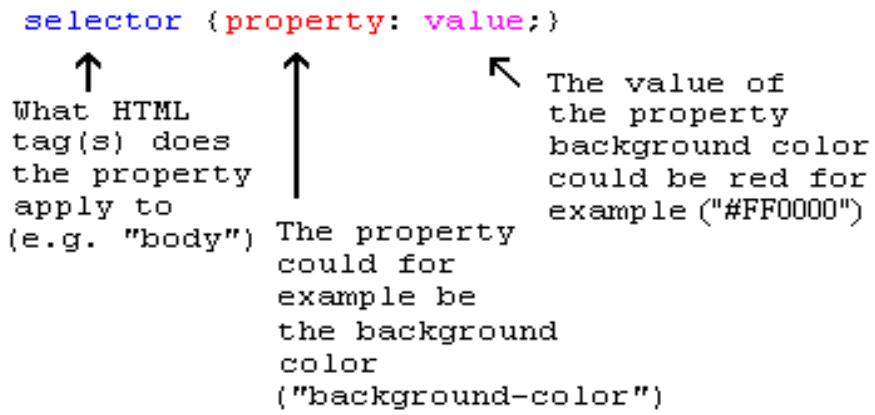
TOPS Technologies

CSS Introduction

- CSS stands for Cascading Style Sheets
- Styles define how to display HTML elements
- Styles were added to HTML 4.0 to solve a problem
- External Style Sheets can save a lot of work
- External Style Sheets are stored in CSS files.
- **CSS Syntax**
  - With html

    <body bgcolor="#FF0000">
  - With CSS

    body {background-color: #FF0000;}



Types Of CSS

- Types of CSS
  - Inline Styles
  - Internal Styles
  - External Styles

Inline (Attribute style)	One way to apply CSS to HTML is by using the HTML attribute style. Make html with the red background color.	<html> <head> <title>Example</title> </head> **<body style="background-color: #FF0000;">** This is a red page </body> </html>

Internal (the tag style)	Another way is to include the CSS codes using the HTML tag <style>.	<html> <head> <title>Example</title> **<style type="text/css">** **body** **{** **background-color: #FF0000;** **}** **</style>** </head> <body> This is a red page </body></html>
External (link to a style sheet)	The recommended method is to link to a so-called external style sheet. An external style sheet is simply a text file with the extension .css. Like any other file, you can place the style sheet on your web server or hard disk.	<html> <head> <title>My document</title> **<link rel="stylesheet" type="text/css"** **href="style/style.css" />** </head> <body>

Pseudo classes

Styles for Special Cases :

Although primarily intended to add styles to particular elements created using HTML tags, there are several cases where we can use CSS to style content on the page that is not specifically set off by HTML tags or to create a dynamic style in reaction to something that your Web site visitor is doing on the screen. These are known as pseudo-elements and pseudo-classes:

1. **Link pseudo-classes:** Used to style hypertext links. Although primarily associated with color, you can actually use any CSS property to set off links and
associated with color, you can actually use any CSS property to set off links and
provide user feedback during interaction.
2. **Dynamic pseudo-classes:** Used to style any element on the screen depending on how the user is interacting with it.
3. **Pseudo-elements:** Used to style the first letter or first line in a block of text.

- Link States:
- All hypertext links have four “states” that can be styled in reaction to a user action.

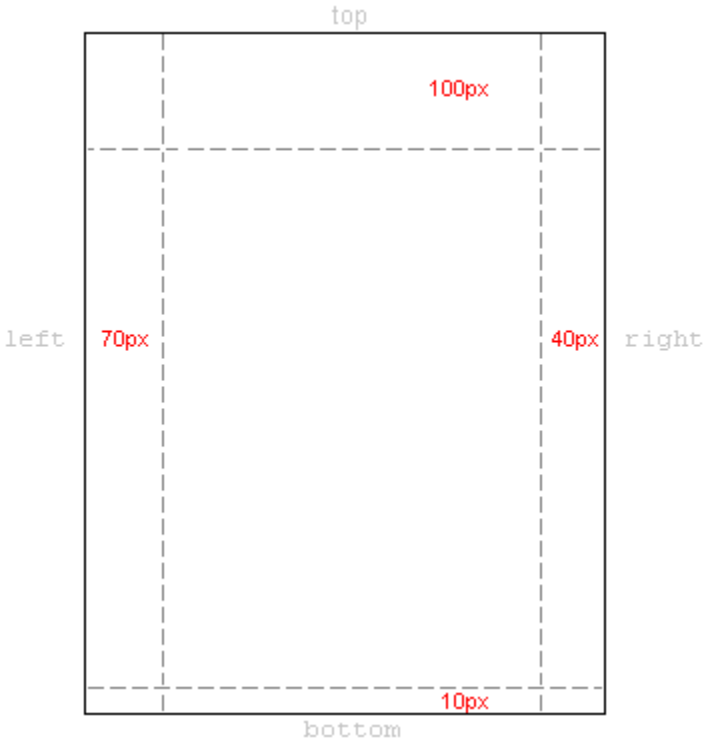
a link state when there has been no action.	a:link {color:grey;} /* unvisited link */
A hover state when the mouse cursor is over it.	a:visited {color:red;} /* visited link */
An active state when the user clicks it.	a:hover {color:yellow;} /* mouse over link */
A visited state when the user returns after having visited the linked-to .	a:active {color:blue;} /* selected link */

Margins and Padding

- Margin and Padding are the two most commonly used properties for spacing-out elements.
- A margin is the space outside something, whereas padding is the space inside something.

Margin

```
body {
        margin-top: 100px;
        margin-right: 40px;
        margin-bottom: 10px;
        margin-left: 70px;
}
OR
body {
        margin: 100px 40px 10px 70px;
}
```



Property	Description
margin	A shorthand property for setting the margin properties in one declaration
margin-bottom	Sets the bottom margin of an element
margin-left	Sets the left margin of an element
margin-right	Sets the right margin of an element
margin-top	Sets the top margin of an element

Padding

```
h1 {
        background: yellow;
        padding: 20px 20px 20px
80px;
}
h2 {
        background: orange;
        padding-left:120px;
}
```

CSS Background

- CSS background properties are used to define the background effects of an element.

		Example
background-color	Sets the background color of an element	background-color:yellow;
background-image	Sets the background image for an element	background-image:url('paper.gif');
background-repeat	Sets how a background image will be repeated.(By default, the background-image property repeats an image both horizontally and vertically.)	background-repeat:repeat-y;(repeat only vertically)

CSS using ID and Class

- Setting a style for a HTML element, CSS allows you to specify your own selectors called "id" and "class".

Id	class
Use id to identify elements that there will only be a single instance of on a page.	Use class to group elements that all behave a certain way.
an ID selector is a name preceded by a **hash character** (“#”).	a class selector is a name preceded by a **full stop** (“.”)
#top { background-color: #ccc; padding: 20px }	.intro { color: red; font-weight: bold; }

```
<div id="top">
<h1>Chocolate curry</h1>
<p class="intro">This is my recipe for making curry purely with chocolate</p>
<p class="intro">Mmm mm mmmmm</p> </div>
```

JavaScript

- Events
- Validations
- Validations with Regular Expression

Events

Events	
onBlur	User has left the focus of the object. For example, they clicked away from a text field that was previously selected.
onChange	User has changed the object, then attempts to leave that fied.
onClick	User clicked on the object.
onDblClick	User clicked twice on the object.
onFocus	User brought the focus to the object.
onKeydown	A key was pressed over an element.
onKeyup	A key was released over an element.
onKeypress	A key was pressed over an element then released
onLoad	The object has loaded.

Mouse Event:	
onMousedown	The cursor moved over the object and mouse/pointing device was pressed down.
onMouseup	The mouse/pointing device was released after being pressed down.
onMouseover	The cursor moved over the object.
onMousemove	The cursor moved while hovering over an object.
onMouseout	The cursor moved off the object.

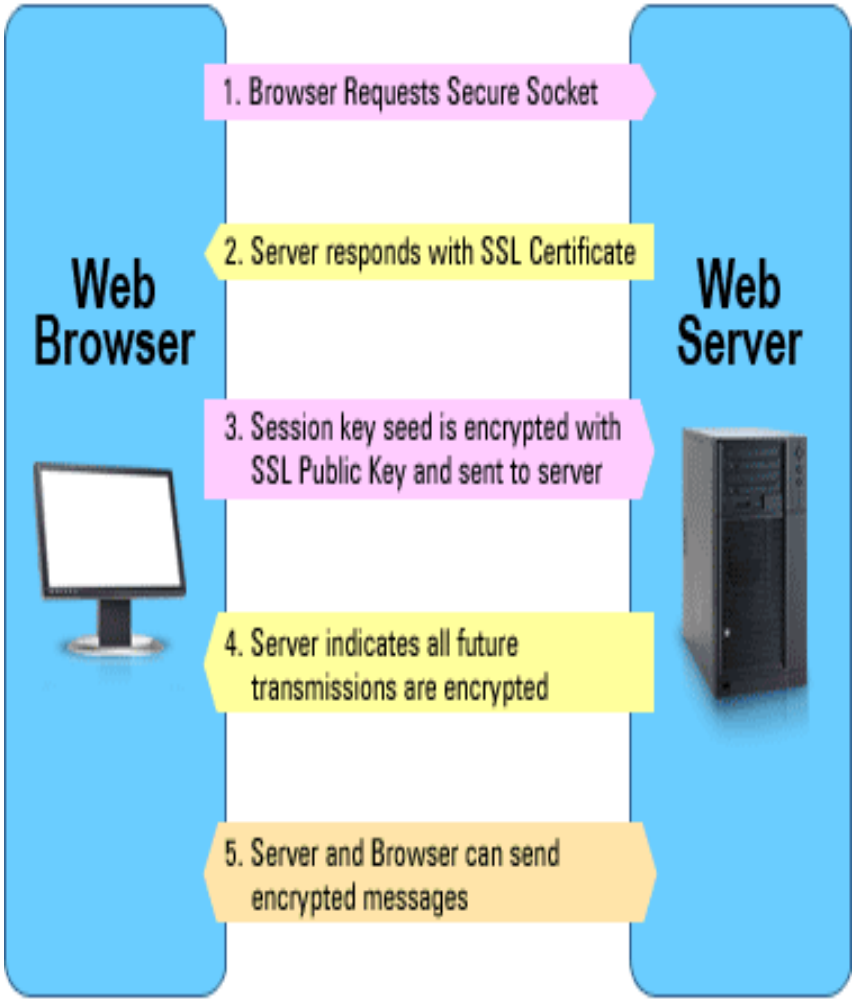
Java Script Validation

Name validation:
Password validation
Email Validation:
Date validation:
Phone Number Validation:
Drop Down Validation
Check Box Validation
Radio Button Validation:

Intro. Client Server Architecture

- The type of computing system in which one powerful workstation serves the requests of other systems, is an example of client server technology. A computer network is an interconnection of computers which share various resources.
- Clients are the individual components which are connected in a network. They have a basic configuration. Client sends a request/query to server and server responds accordingly. Please note that the client doesn't share any of its resources. They are subordinates to servers, and their access rights are defined by servers only. They have localized databases.
- The **client–server model** is a distributed application structure in computing that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server is a host that is running one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests.

Web Browser and Web Server



Web Browser

- A web browser is a software application for retrieving, presenting, and traversing information resources on the World Wide Web. An information resource is identified by a Uniform Resource Identifier (URI) and may be a web page, image, video, or other piece of content. Hyperlinks present in resources enable users easily to navigate their browsers to related resources. A web browser can also be defined as an application software or program designed to enable users to access, retrieve and view documents and other resources on the Internet.
- Example : Mozila Fire Fox, Google Crome, Safari, IE, Netscape Browser

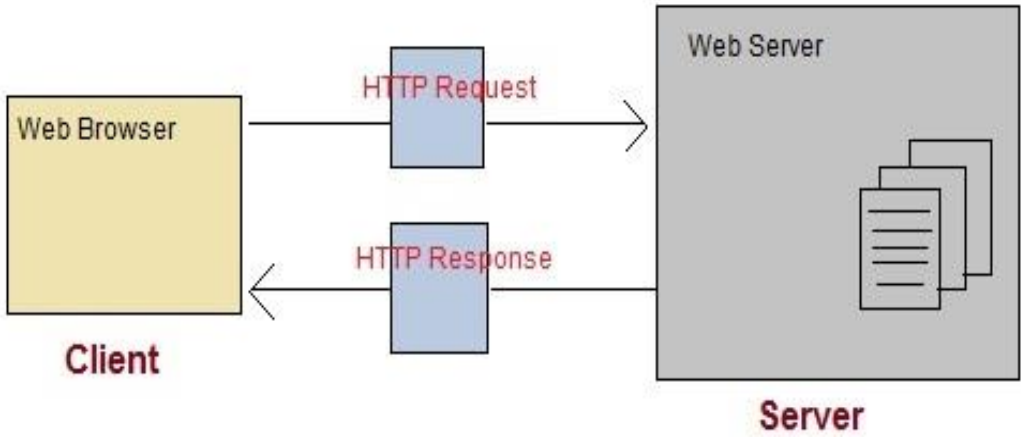
Web Server

- Web server can refer to either the hardware (the computer) or the software (the computer application) that helps to deliver content that can be accessed through the Internet.
- Example: Apache tomcat, Jboss, Glass Fish, Web Sphere, Netscape Navigator Server(PHP), IIS (Windows)

HTTP(Hyper Text Transfer Protocol)

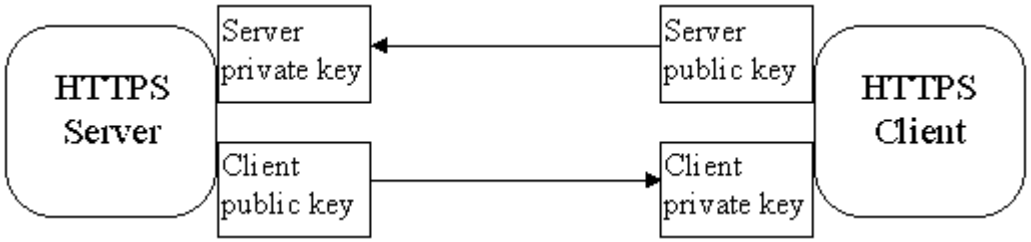
- **The Hypertext Transfer Protocol** (**HTTP**) is an application protocol for distributed, collaborative, hypermedia information systems.HTTP is the foundation of data communication for the World Wide Web.
- Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

- Short for **H***yper***T***ext* **T***ransfer* **P***rotocol*, the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page.



HTTPS(Hypertext Transfer Protocol Secure)

- **Hypertext Transfer Protocol Secure** (**HTTPS**) is a communications protocol for secure communication over a computer network, with especially wide deployment on the Internet.
- Hyper Text Transfer Protocol Secure (HTTPS) is a secure version of the Hyper Text Transfer Protocol (http). HTTPS allows secure ecommerce transactions, such as online banking.
- Web browsers such as Internet Explorer and Firefox display a padlock icon to indicate that the website is secure, as it also displays https:// in the address bar.



HTTP Overview

- **HTTP is connectionless:** The HTTP client ie. browser initiates an HTTP request and after a request is made, the client disconnects from the server and waits for a response. The server process the request and re-establish the connection with the client to send response back.
- **HTTP is media independent:**This means, any type of data can be sent by HTTP as long as both the client and server know how to handle the data content. This is required for client as well as server to specify the content type using appropriate MIME-type.
- **HTTP is stateless:** As mentioned above, HTTP is a connectionless and this is a direct result that HTTP is a stateless protocol. The server and client are aware of each other only during a current request. Afterwards, both of them forget about each other. Due to this nature of the protocol, neither the client nor the browser can retain information between different request across the web pages.
- HTTP makes use of the Uniform Resource Identifier (URI) to identify a given resource and to establish a connection. Once connection is established, messages are passed in a format similar to that used by Internet mail [RFC5322] and the Multipurpose Internet Mail Extensions (MIME) [RFC2045].

The GET Method

Note that query strings (name/value pairs) is sent in the URL of a GET request:

/test/demo_form.asp**?name1=value1&name2=value2**

- **GET requests:**
  - GET requests can be cached
  - GET requests remain in the browser history
  - GET requests can be bookmarked
  - GET requests should never be used when dealing with sensitive data
  - GET requests have length restrictions
  - GET requests should be used only to retrieve data

The Post Method

Note that query strings (name/value pairs) is sent in the HTTP message body of a POST request:

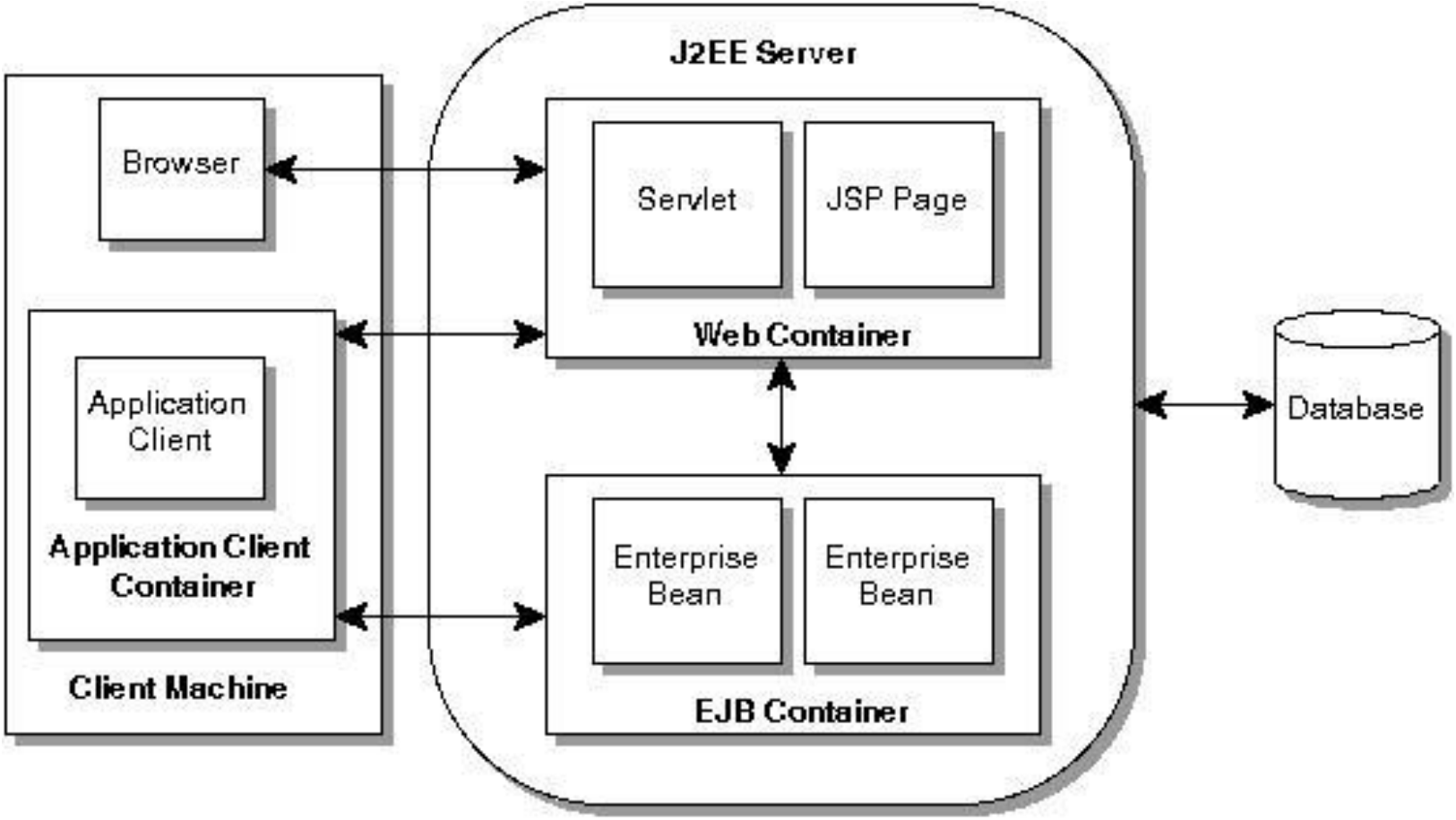
POST /test/demo_form.asp HTTP/1.1

Host: w3schools.com

name1=value1&name2=value2

- **Some other notes on POST requests:**
  - POST requests are never cached
  - POST requests do not remain in the browser history
  - POST requests cannot be bookmarked
  - POST requests have no restrictions on data length

J2EE Architecture



- **J2EE Server**
  - The J2EE server provides the following services:
  - Naming and Directory - allows programs to locate services and components through the Java Naming and Directory Interface(JNDI) API
  - Authentication - enforces security by requiring users to log in
  - HTTP - enables Web browsers to access servlets and Java Server Pages(JSP) files
  - EJB - allows clients to invoke methods on enterprise beans
- **Web Container**
- The Web container is a runtime environment for JSP files and servlets. Although these Web components are an important part of a J2EE application, this manual focuses on enterprise beans. For more information on developing Web components, see the home pages for the Java Server Pages and Servlet technologies.

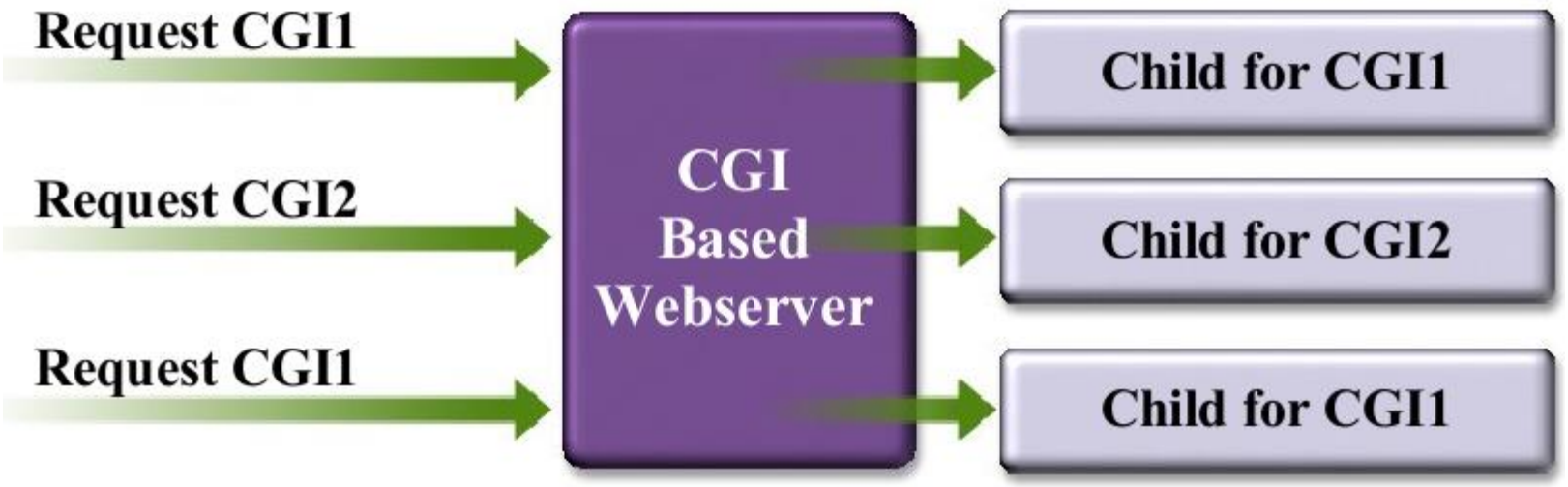
- Modular design allows for easier modification of the business logic.
- Enterprise components can use container-provided services such as security, transaction, persistence, and life cycle management.

- **EJB Container**
- Enterprise bean instances run within an EJB container. The container is a runtime environment that controls the enterprise beans and provides them with important system-level services. Since you don't have to develop these services yourself, you are free to concentrate on the business methods in the enterprise beans.

Common Gateway Interface(CGI)

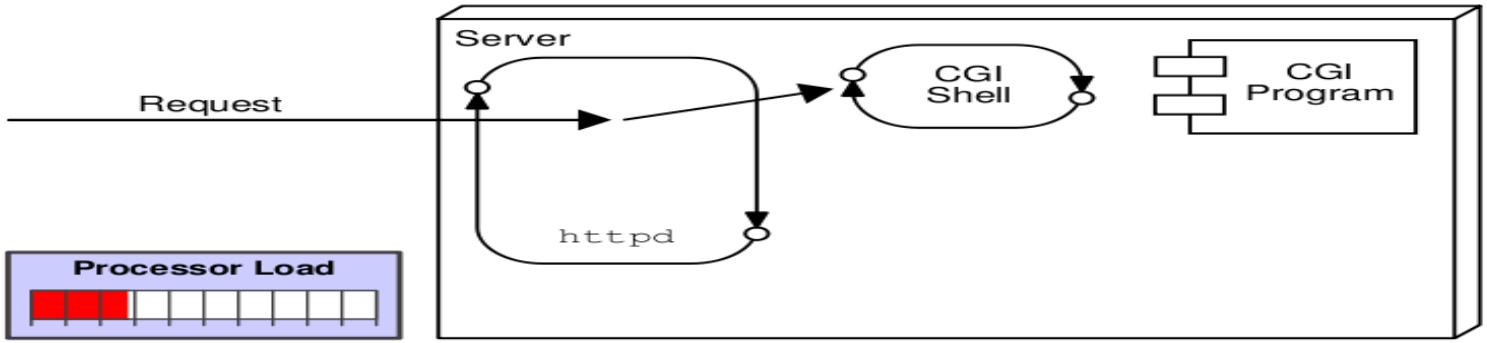
- The CGI connects Web servers to external applications.
- CGI is the interface between server programs and other software.
- CGI can do two things.
  - It can gather information sent from a web browser to a web server, and make the information available to an external program.
  - CGI can send the output of a program to a Web browser that request it.

CGI Process

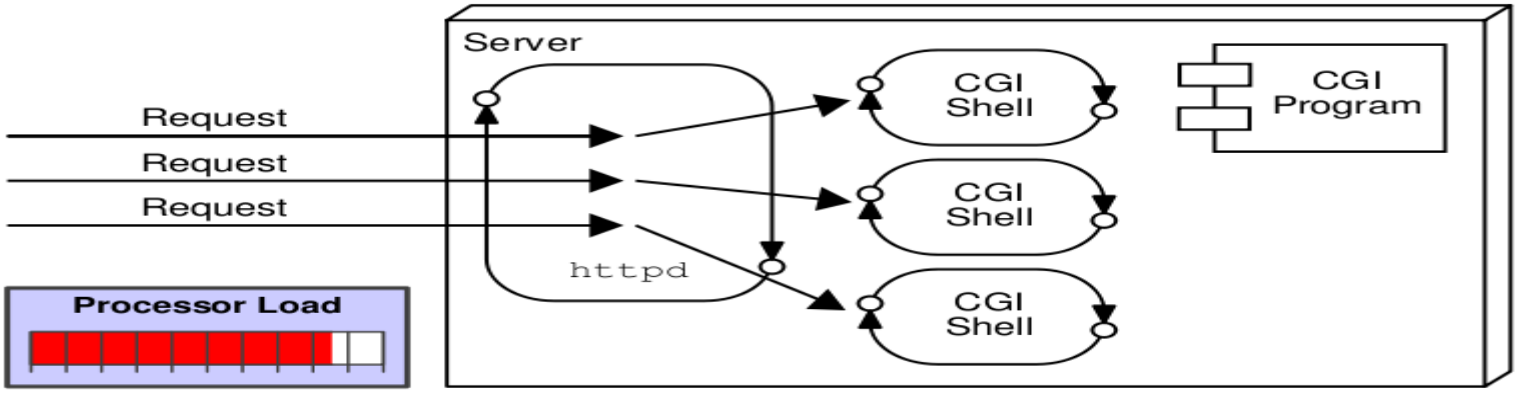


Execution of CGI Programs

How CGI works with one request:



How CGI works with many requests:



Advantages and Disadvantages of CGI Programs

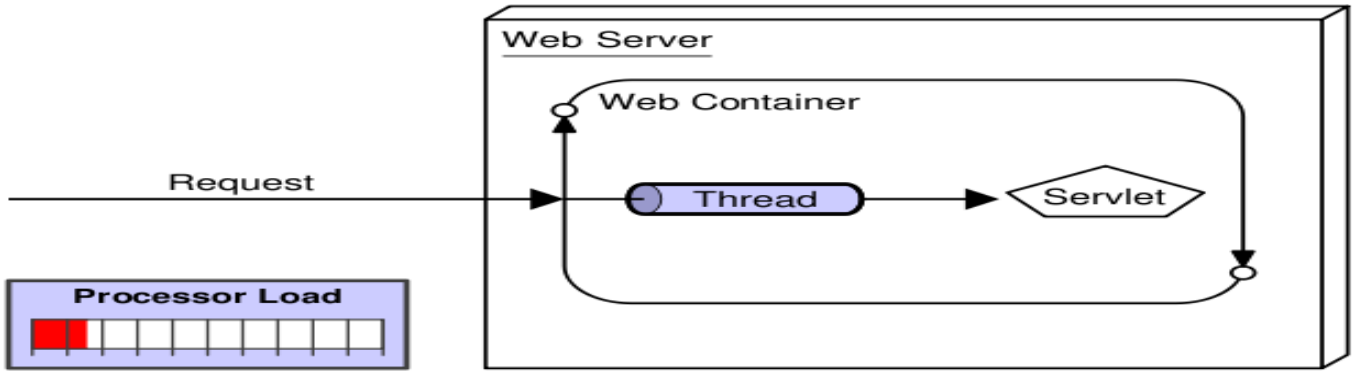
- CGI Advantages
  - Written in a variety of languages
  - Relatively easy for a web designer to reference
- CGI Disadvantages
  - If number of clients increases, it takes more time for sending response.
  - For each request, it starts a process and Web server is limited to start processes.
  - It uses platform dependent language e.g. C, C++, perl.

Servlet Programming Process

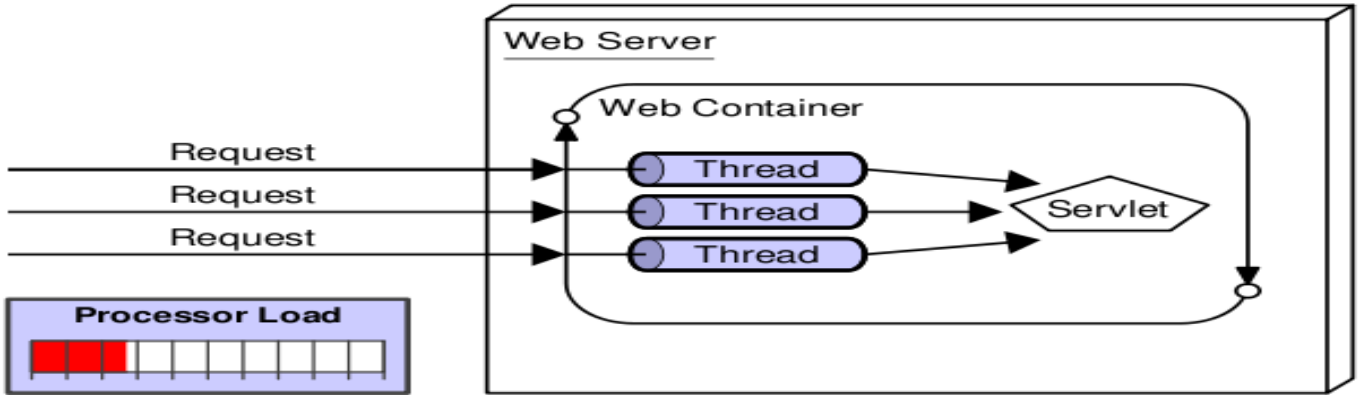
- A **servlet** is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes.
- The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets. All servlets must implement the Servlet interface, which defines life-cycle methods. When implementing a generic service, you can use or extend the GenericServlet class provided with the Java Servlet API. The HttpServlet class provides methods, such as doGet and doPost, for handling HTTP-specific services.
- This chapter focuses on writing servlets that generate responses to HTTP requests.

Execution of Servlet Programs

How servlets work with one request:



How servlets work with many requests:



CGI vs Servlet

CGI	Servlet
Written in C, C++, Visual Basic and Perl	Written in Java
Difficult to maintain, non-scalable, non- manageable	Powerful, reliable, and efficient
Prone to security problems of programming language	Improves scalability, reusability (component based)
Resource intensive and inefficient	Leverages built-in security of Java programming language
Platform and application-specific	Platform independent and portable

Servlet Advantages Over CGI

- There are many advantages of Servlet over CGI. The web container creates threads for handling the multiple requests to the servlet. Threads have a lot of benefits over the Processes such as they share a common memory area, lighweight, cost of communication between the threads are low. The basic benefits of servlet are as follows:
  - **Better performance:** because it creates a thread for each request not process.
  - **Portability:** because it uses java language.
  - **Robust:** Servlets are managed by JVM so no need to worry about momory leak, garbage collection etc.
  - **Secure:** because it uses java language..

Example :
https://github.com/TopsCode/Java/tree/master/Module-4/4.0.1.ServletPractical

Servlets

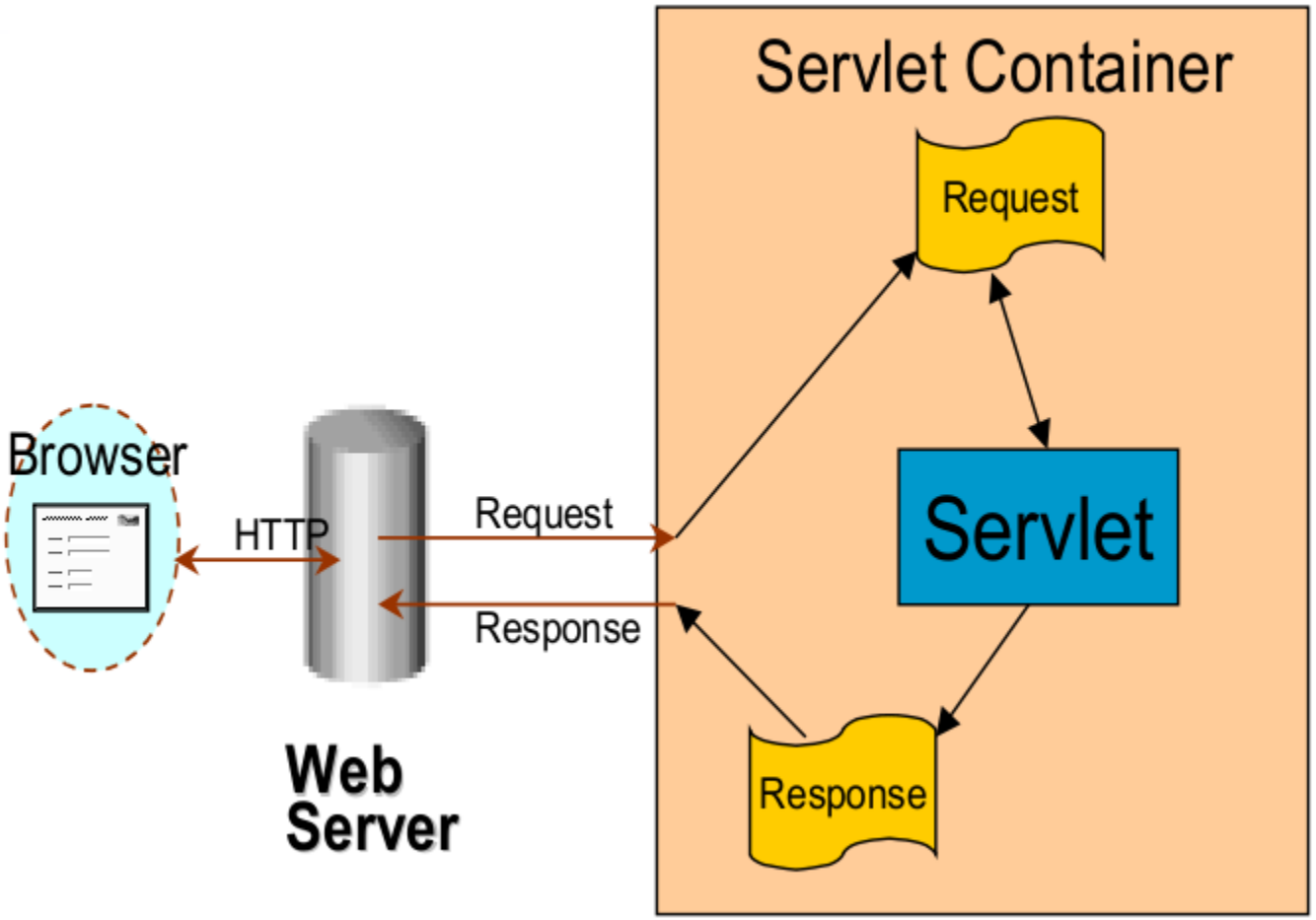
- Introducation and Task
- Servlet Versions
- Servlet API
- Types of Servlets.
- Differance between HTTPServlet and GenricServlet
- Servlet Life Cycle
- Creating Servlets
- Servlet Entry in Web.xml file
- Logical URL
- Request Dispatcher Interface
- Servlet Config Interface
- Servlet Context Interface
- WebApplication Listener
- Servlet Attributes

Introducation of Servlet

- **Servlet** technology is used to create web application (resides at server side and generates dynamic web page).
- **Servet** technology is robust and scalable as it uses the java language. Before Servlet, CGI (Common Gateway Interface) scripting language was used as a server-side programming language. But there were many disadvantages of this technology. We have discussed these disadvantages below.
- There are many interfaces and classes in the servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse etc.

- Servlet can be described in many ways, depending on the context.
  - Servlet is a technology i.e. used to create web application.
  - Servlet is an API that provides many interfaces and classes including documentations.
  - Servlet is an interface that must be implemented for creating any servlet.
  - Servlet is a class that extend the capabilities of the servers and respond to the incoming request. It can respond to any type of requests.
  - Servlet is a web component that is deployed on the server to create dynamic web page.

Http Servlet Request Response



Servlet Task

- Read the explicit data sent by the clients (browsers). This includes an HTML form on a Web page or it could also come from an applet or a custom HTTP client program.
- Read the implicit HTTP request data sent by the clients (browsers). This includes cookies, media types and compression schemes the browser understands, and so forth.
- Process the data and generate the results. This process may require talking to a database, executing an RMI or CORBA call, invoking a Web service, or computing the response directly.

- Send the explicit data (i.e., the document) to the clients (browsers). This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), Excel, etc.
- Send the implicit HTTP response to the clients (browsers). This includes telling the browsers or other clients what type of document is being returned (e.g., HTML), setting cookies and caching parameters, and other such tasks.

Servlet Advantage and Disadvantage

- **Servlet Advantage**

  1. Servlets provide a way to generate dynamic documents that is both easier to write and faster to run.

  2. provide all the powerfull features of JAVA, such as Exception handling and garbage collection.

  3. Servlet enables easy portability across Web Servers.

  4. Servlet can communicate with different servlet and servers.

  5. Since all web applications are stateless protocol, servlet uses its own API to maintain session

- **Servlet Disadvantage**

  1. Designing in servlet is difficult and slows down the application.

  2. Writing complex business logic makes the application difficult to understand.

  3. You need a Java Runtime Environment on the server to run servlets. CGI is a completely language independent protocol, so you can write CGIs in whatever languages you have available (including Java if you want to).

Servlet Versions

Servlet API version	Released	Specification	Platform	Important Changes
Servlet 4.0	Sep 2017	JSR 369	Java EE 8	HTTP/2
Servlet 3.1	May 2013	JSR 340	Java EE 7	Non-blocking I/O, HTTP protocol upgrade mechanism (WebSocket)[14]
Servlet 3.0	December 2009	JSR 315	Java EE 6, Java SE 6	Pluggability, Ease of development, Async Servlet, Security, File Uploading
Servlet 2.5	September 2005	JSR 154	Java EE 5, Java SE 5	Requires Java SE 5, supports annotation
Servlet 2.4	November 2003	JSR 154	J2EE 1.4, J2SE 1.3	web.xml uses XML Schema
Servlet 2.3	August 2001	JSR 53	J2EE 1.3, J2SE 1.2	Addition of `Filter`
Servlet 2.2	August 1999	JSR 902, JSR 903	J2EE 1.2, J2SE 1.2	Becomes part of J2EE, introduced independent web applications in .war files
Servlet 2.1	November 1998	2.1a	Unspecified	First official specification, added `RequestDispatcher`, `ServletContext`
Servlet 2.0	December 1997	N/A	JDK 1.1	Part of April 1998 Java Servlet Development Kit 2.0[15]
Servlet 1.0	December 1996	N/A		Part of June 1997 Java Servlet Development Kit (JSDK) 1.0[9]

Types of Servlets

- There is two types
  - Generic Servlet
    - Generic servlet is protocol independent servlet. It implements the Servlet and ServletConfig interface. It may be directly extended by the servlet. Writing a servlet in in GenericServlet is very easy. It has only init() and destroy() method of ServletConfig interface in its life cycle. It also implements the log method of ServletContext interface.
  - Http Servlet
    - HttpServlet is HTTP (Hyper Text Transfer Protocol) specific servlet. It provides an abstract class HttpServlet for the developers for extend to create there own HTTP specific servlets. It provides http specific methods such as doGet, doPost, doHead, doTrace etc.

Generic Servlet

- **javax.servlet.GenericServlet**
  Signature: public abstract class GenericServlet extends java.lang.Object implements Servlet, ServletConfig, java.io.Serializable
  - GenericServlet defines a generic, protocol-independent servlet.
  - GenericServlet gives a blueprint and makes writing servlet easier.
  - GenericServlet provides simple versions of the lifecycle methods init and destroy and of the methods in the ServletConfig interface.
  - GenericServlet implements the log method, declared in the ServletContext interface.
  - To write a generic servlet, it is sufficient to override the abstract service method.
    - **Example :**
    - https://github.com/TopsCode/Java/blob/master/Module-4/4.0.1.ServletPractical/src/com/controller/ClassWithServletInterface.java

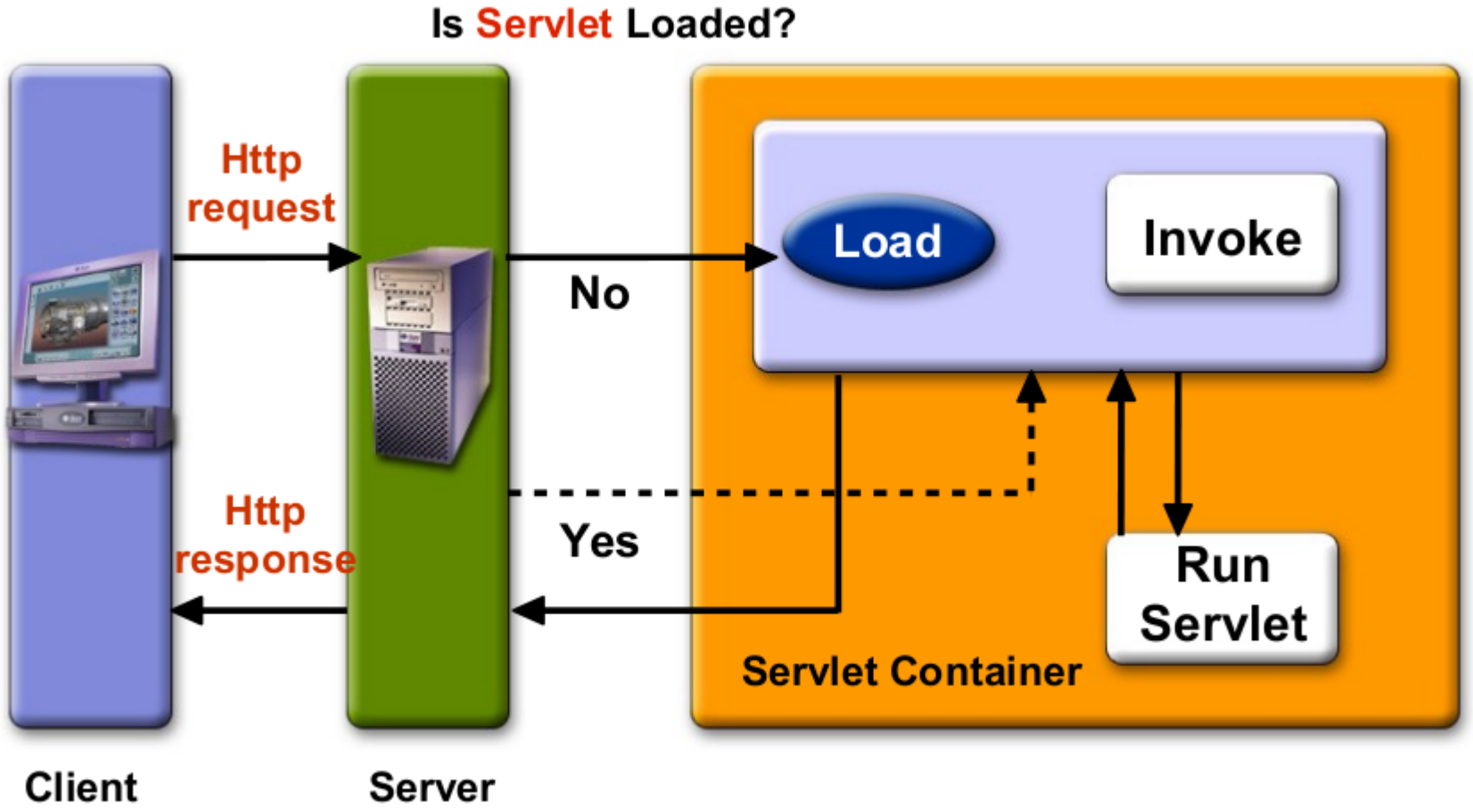
Http Servlet

- **javax.servlet.http.HttpServlet**
  Signature: public abstract class HttpServlet extends GenericServlet implements java.io.Serializable
  - HttpServlet defines a HTTP protocol specific servlet.
  - HttpServlet gives a blueprint for Http servlet and makes writing them easier.
  - HttpServlet extends the GenericServlet and hence inherits the properties GenericServlet.
  - **Example:**
  - https://github.com/TopsCode/Java/blob/master/Module-4/4.0.1.ServletPractical/src/com/controller/HttpServletDemo.java

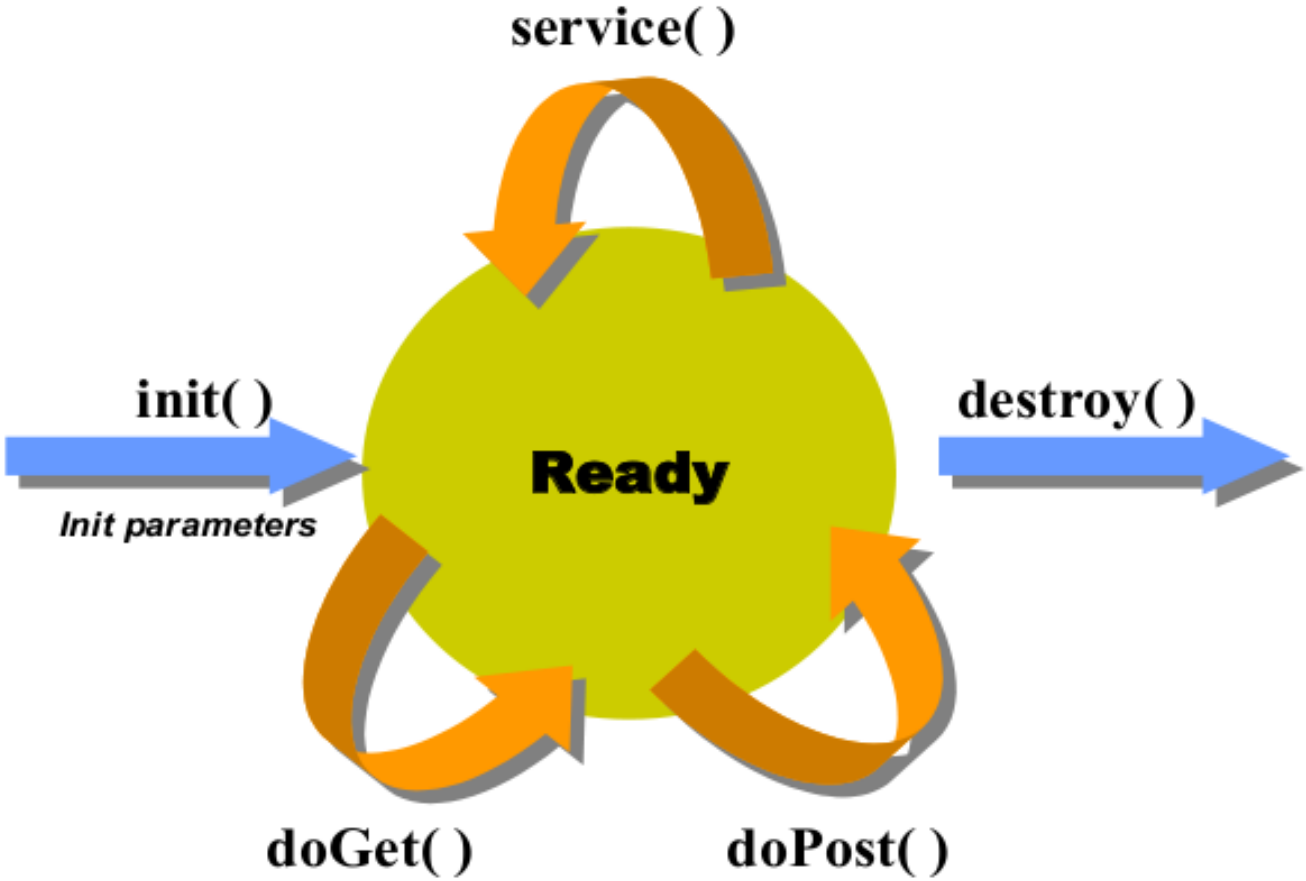
Difference between HttpServlet and GenericServlet

Generic Servlet	Http Servlet
GenericServlet class is direct subclass of Servlet interface.	HttpServlet class is the direct subclass of Generic Servlet.
Generic Servlet is protocol independent.It handles all types of protocol like http, smtp, ftp etc.	HttpServlet is protocol dependent. It handles only http protocol.
Generic Servlet only supports service() method.It handles only simple request public void service(ServletRequest req,ServletResponse res).	HttpServlet supports public void service(ServletRequest req,ServletResponse res) and protected void service(HttpServletRequest req,HttpServletResponse res).
Generic Servlet only supports service() method.	HttpServlet supports also doGet(),doPost(),doPut(),doDelete(),doHead(),doTrace(),doOptions()etc.

Servlet Request Response Process



Servlet Life Cycle Methods



- A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet
  - The servlet is initialized by calling the **init ()** method.
  - The servlet calls **service()** method to process a client's request.
  - The servlet is terminated by calling the **destroy()** method.
  - Finally, servlet is garbage collected by the garbage collector of the JVM.

Example :

https://github.com/topscode/java/blob/master/module-4/4.0.1.servletpractical/src/com/controller/lifecycleofservlet.java

Logical URL

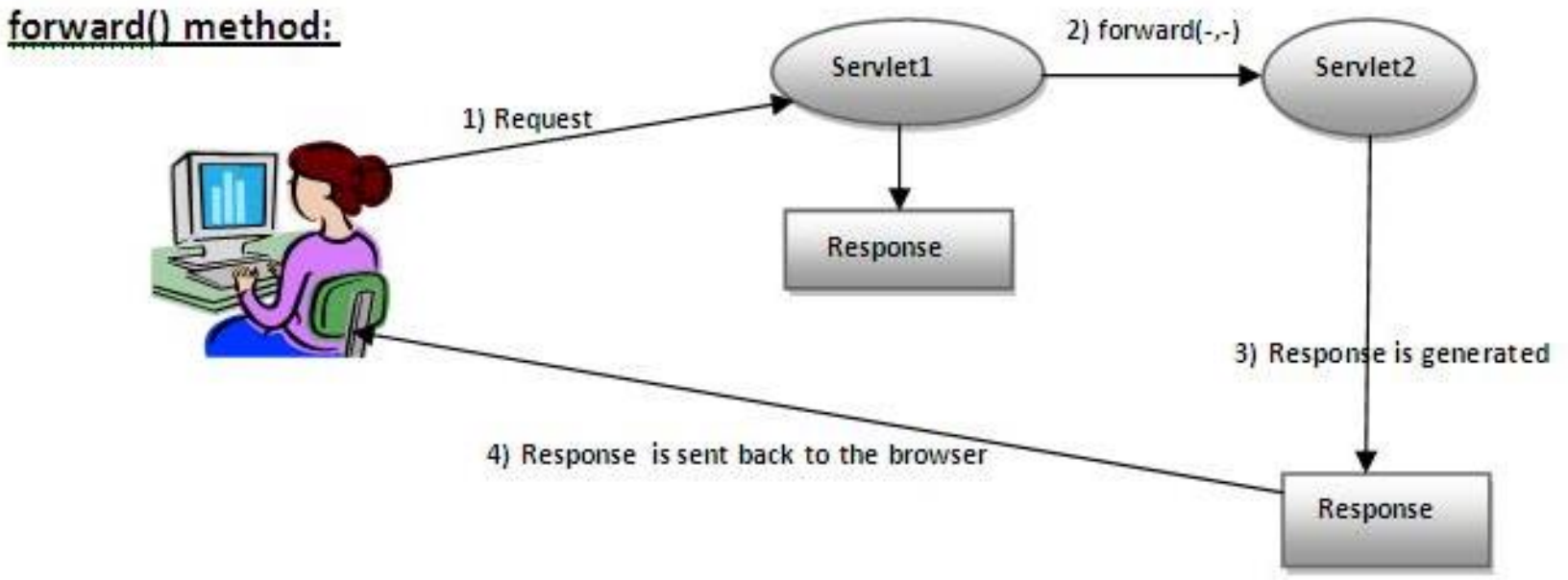
- The client queries a name server to get the list of physical URLs corresponding to a given logical URL. The name service can be either independent of the Web servers or some of the Web servers can also act as name servers. A scheme based on independent name servers similar to the DNS service was proposed for binding of Uniform Resource Names (URNs) to IP addresses
- The server looks up all lists of physical URLs corresponding to every logical URL that occurs in a requested HTML resource. The list is piggybacked on the response sent to the client.

Example :
https://github.com/TopsCode/Java/blob/master/Module-4/4.0.1.ServletPractical/src/com/controller/LifeCycleOfServlet.java

Request Dispatcher

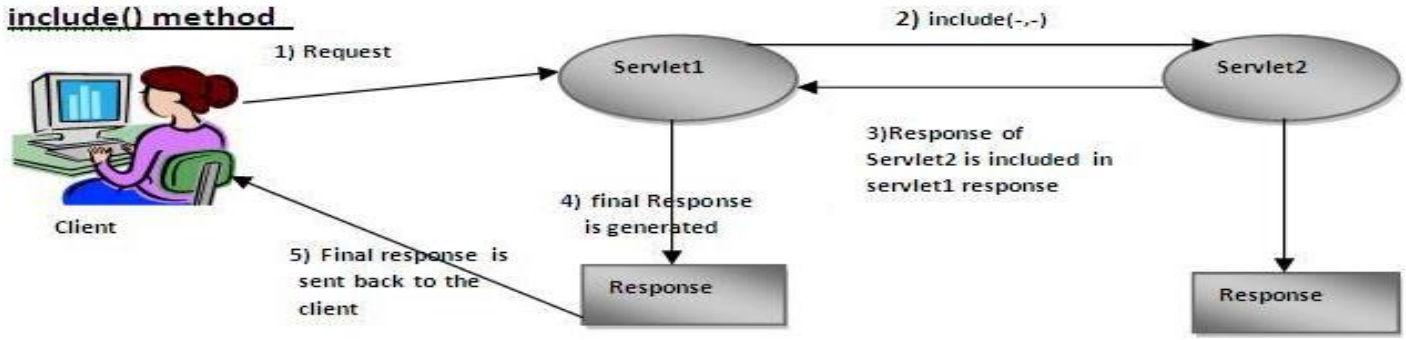
- The RequestDispacher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp.This interface can also be used to include the content of antoher resource also. It is one of the way of servlet collaboration.
- There are two methods defined in the RequestDispatcher interface.
  - **public void forward(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
  - **public void include(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Includes the content of a resource (servlet, JSP page, or HTML file) in the response.

Forword Method



- As you see in the above figure, response of second servlet is sent to the client. Response of the first servlet is not displayed to the user.

Include Method



- As you can see in the above figure, response of second servlet is included in the response of the first servlet that is being sent to the client.

Example : https://github.com/topscode/java/tree/master/module-4/4.3filterdemo

Send Redirect

- The sendRedirect() method of HttpServletResponse interface can be used to redirect response to another resource, it may be servlet, jsp or html file. It can accept relative URL. This method can work inside and outside the server as it uses the URL bar of the browser.
- public void sendRedirect(String URL)throws IOException;

Example :

https://github.com/TopsCode/Java/tree/master/Module-4/4.0.2CrudOperation/src/com/Controller

Servlet Config

- An object of ServletConfig is created by the web container for each servlet. This object can be used to get configuration information from web.xml file.
- If the configuration information is modified from the web.xml file, we don't need to change the servlet. So it is easier to manage the web application if any specific content is modified from time to time.
- 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.

- **Methods of ServletConfig interface**
  - **public String getInitParameter(String name):**Returns the parameter value for the specified parameter name.
  - **public Enumeration getInitParameterNames():**Returns an enumeration of all the initialization parameter names.
  - **public String getServletName():**Returns the name of the servlet.
  - **public ServletContext getServletContext():**Returns an object of ServletContext.

Example :

https://github.com/TopsCode/Java/tree/master/Module-4/4.1ServletConfigContext

Servlet Context

- An object of ServletContext is created by the web container at time of deploying the project. This object can be used to get configuration information from web.xml file. There is only one ServletContext object per web application.
- If any information is shared to many servlet, it is better to provide it from the web.xml file using the **<context-param>** element.
- The **advantage of Servlet Context** Easy to maintain if any information is shared to all the servlet, it is better to make it available for all the servlet. We provide this information from the web.xml file, so if the information is changed, we don't need to modify the servlet. Thus it removes maintenance problem.

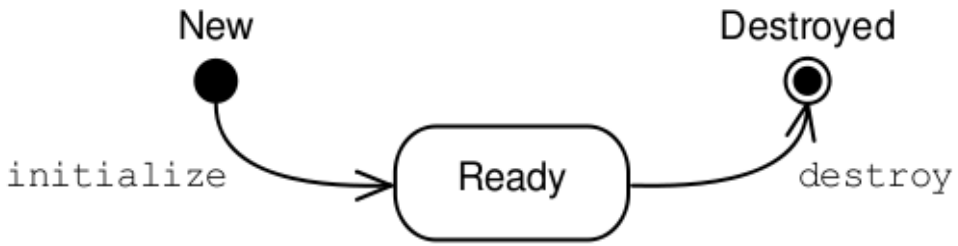
- **Methods of ServletContext interface**
  - **public String getInitParameter(String name):**Returns the parameter value for the specified parameter name.
  - **public Enumeration getInitParameterNames():**Returns the names of the context's initialization parameters.
  - **public void setAttribute(String name,Object object):**sets the given object in the application scope.
  - **public Object getAttribute(String name):**Returns the attribute for the specified name.
  - **public Enumeration getInitParameterNames():**Returns the names of the context's initialization parameters as an Enumeration of String objects.
  - **public void removeAttribute(String name):**Removes the attribute with the given name from the servlet context.

Difference between Context and Config

Servlet Context	Servlet Config
One per Web Application	One per Servlet
Use it to across web app parameter (Confirmed in Deployment Descriptor)	Use it to pass deploy-time parameters to the servlet (i.e. database, filename, etc.)
Use for internal application communication btw servlets, JSPs, and other components	Use to across the Servlet Context
Used to across server information about the container and the version of the API it supports	Parameters are configured in the Deployment Descriptor

Web Application Listener

- Events are basically occurrence of something. Changing the state of an object is known as an event. We can perform some important tasks at the occurrence of these exceptions, such as counting total and current logged-in users, creating tables of the database at time of deploying the project, creating database connection object etc. There are many Event classes and Listener interfaces in the javax.servlet and javax.servlet.http packages.
- When the web container is started, each web application is initialized.
- When the web container is shutdown, each web application is destroyed.
- A servlet context listener can be used to receive these web application life cycle events.



Web Application Listener

There are two levels of servlet events:

- Servlet Context-level (application-level) event
  - This event involves resources or state held at the level of the application servlet context object.
- Session-level event
  - This event involves resources or state associated with the series of requests from a single user session; that is, associated with the HTTP session object.

Each of these two levels has two event categories:

- Lifecycle changes
- Attribute changes

ServletContextListener is a interface which contains two methods:

- **public void contextInitialized(ServletContextEvent event)**
- **public void contextDestroyed(ServletContextEvent event)**

When we implement any interface then we have to implement its all methods. This listener will help a application to start and shutdown the events.

How the ServletContextListener is useful:

1. ServletContextListener is notified when the context is initialized.
   a). ServletContextListener gets the context init parameters from the ServletContext.
   b). It stores the database connection as an attribute, so that the other components in the web application can access it.
2. It will be notified when the context is destroyed. It closes the database connection.

Example :
https://github.com/TopsCode/Java/tree/master/Module-4/4.2WebListenerDemo

Attribute in Servlet

- An attribute is an object that can be set, get or removed from one of the following scopes:
  - request scope
  - session scope
  - application scope
- The servlet programmer can pass informations from one servlet to another using attributes. It is just like passing object from one class to another so that we can reuse the same object again and again.
- There are following 4 attribute specific methods. They are as follows:
  - **public void setAttribute(String name,Object object):**sets the given object in the application scope.
  - **public Object getAttribute(String name):**Returns the attribute for the specified name.
  - **public Enumeration getInitParameterNames():**Returns the names of the context's initialization parameters as an Enumeration of String objects.
  - **public void removeAttribute(String name):**Removes the attribute with the given name from the servlet context.

    Example :

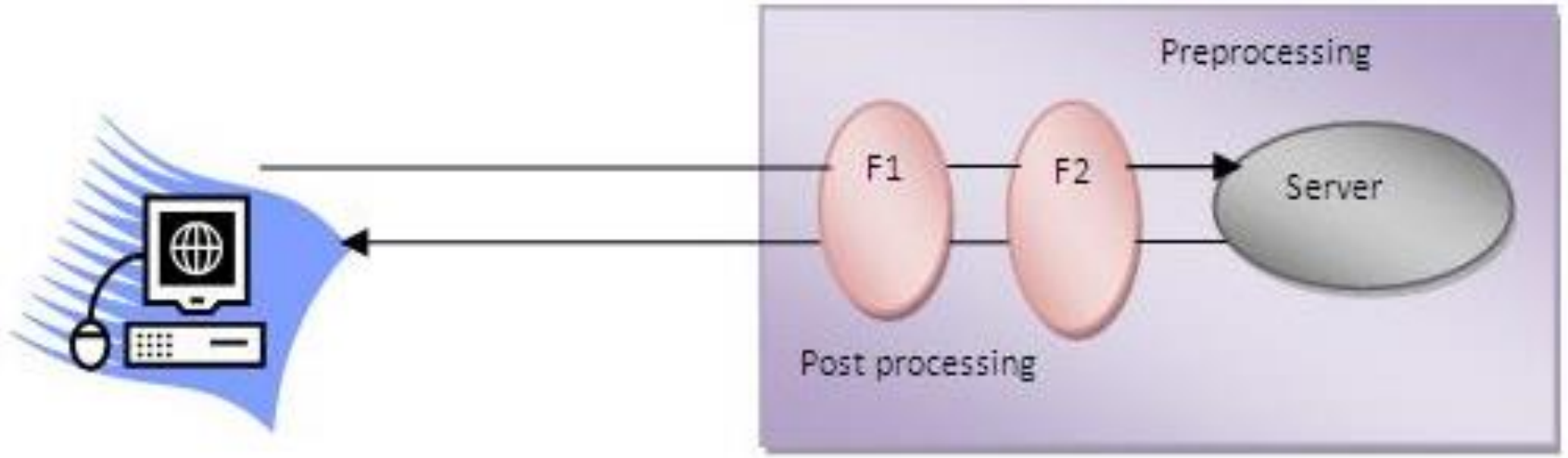
https://github.com/TopsCode/Java/tree/master/Module-4/4.0.2CrudOperation/src/com/Controller

Filter

- Introducation
- What are the needs?
- Process of Execution Filter
- Filter Interface
- FilterChain Interface
- Filter Lifecycle
- Applying Filter
- Entry in Web.xml file
- URL Pattern with Filter.

Introduction

- A filter is an object that is used to perform filtering tasks such as conversion, log maintain, compression, encryption and decryption, input validation etc. A filter is invoked at the preprocessing and postprocessing of a request. It is pluggable, i.e. its entry is defined in the web.xml file, if we remove the entry of filter from the web.xml file, filter will be removed automatically and we don't need to change the servlet. So it will be easier to maintain the web application.



Usage of Filter

- recording all incoming requests
- logs the IP addresses of the computers from which the requests originate
- conversion
- data compression
- encryption and decryption
- input validation etc.

Advantage of Filter

- Filter is pluggable.
- One filter don't have dependency onto another resource.

What are the needs??

- Filters can be used for many activities in a web application, such as:
  - Blocking access to a resource based on user identity or role membership
  - Auditing incoming requests
  - Compressing the response data stream
  - Transforming the response
  - Measuring and logging servlet performance

Example :

https://github.com/TopsCode/Java/tree/master/Module-4/4.3FilterDemo

Filter Life Cycle Methods

Life Cycle Stage	Method	Description
Initialization – init()	**public void init(FilterConfig filterConfig)**	This method is called by the web container to indicate to a filter that it is being placed into service. When filer deploy in web container.
Filter method – doFilter()	**public void doFilter (ServletRequest, ServletResponse, FilterChain)**	• This method is called by the container each time a request/response pair is passed through the chain due to a client request for a resource at the end of the chain.
Destroy – destroy()	**public void destroy()**	This method is called by the web container to indicate to a filter that it is being taken out of service. When filter destroy from web container

JSP

- JSP Introduction
- JSP Translation
- JSP Life Cycle
- Comments
- Directives
- Scriplets

- Expression
- Declration
- Implicit Objects
- Action Tags
- EL
- JSTL
- Custom Tags

Introduction

- **JSP** technology is used to create web application just like Servlet technology. It can be thought of as an extension to the servlet because it provides more functionality than servlet such as expression language, jstl etc.
- A JSP page contains HTML code and JSP tags. The jsp pages are easier to maintain than servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tag etc.
- The goal of JSP technology is to support separation of presentation and business logic:
  - Web designers can design and update pages without learning the Java programming language.
  - Programmers for Java platform can write code without dealing with web page design.

- JSP pages are translated into Java servlet classes, are compiled, and are treated just like servlets in the web container.
- Java Server Pages enable you to write standard HTML pages containing tags that run powerful programs based on Java technology.
- If designed well, JSP pages focus on the presentation logic, not on the business logic.
- In JSP pages, custom tags and JSP Expression Language provide for reusable code and separation of concerns.
- In a Java technology web application, JSP pages are often used in conjunction with servlets and business objects in a Model-View-Controller pattern.

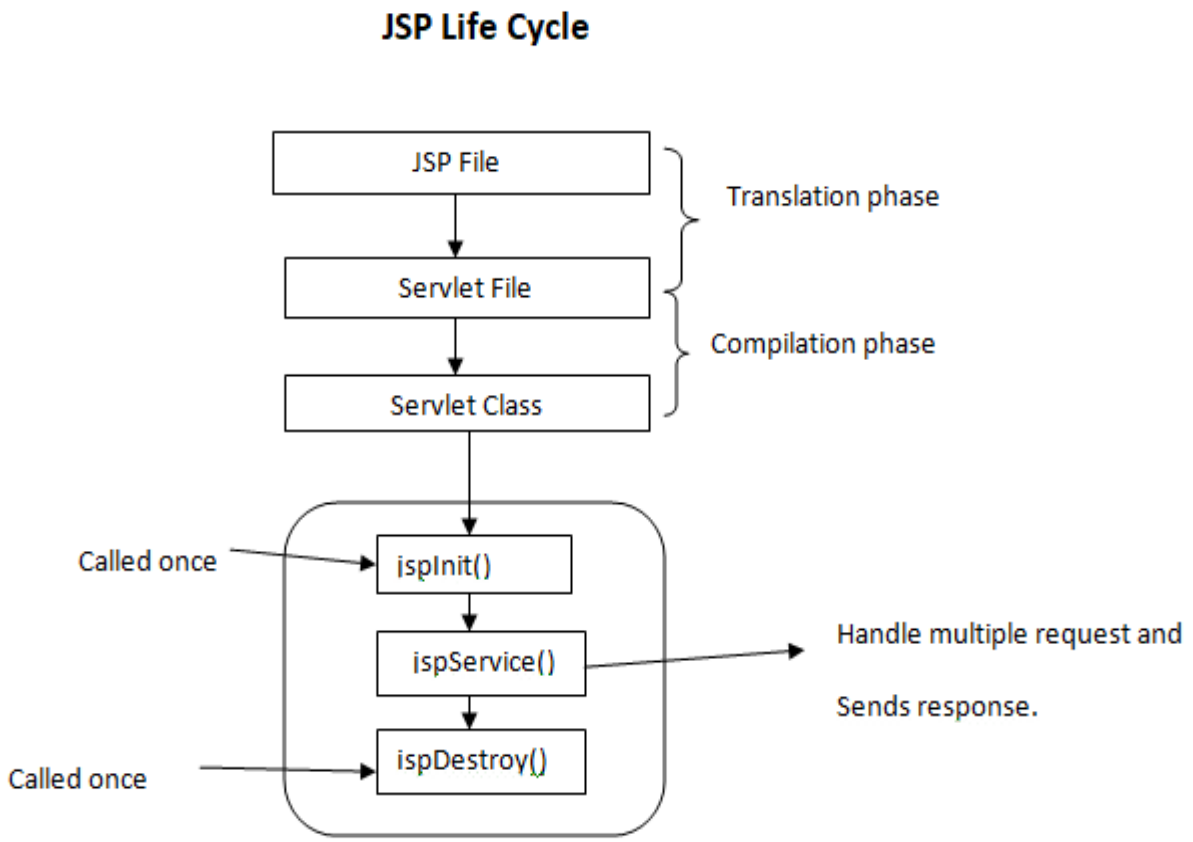
Advantage of JSP over Servlet

- **Extension to Servlet** JSP is the extension to the servlet technology. We can use all the features of Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.
- **Easy to maintain** JSP can be easily managed because we can easily separate our business logic with presentation logic. In servet, we mix our business logic with the presentation logic.
- **Fast Development** If JSP page is modified, we don't need to redeploy the project. The servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

JSP Life Cycle

1. Translation of JSP Page
2. Compilation of JSP Page
3. Classloading (class file is loaded by the classloader)
4. Instantiation (Object of the Generated Servlet is created).
5. Initialization (jspInit() method is invoked by the container).
6. Reqeust processing (_jspService() method is invoked by the container).
7. Destroy (jspDestroy() method is invoked by the container).

JSP Life Cycle with Methods



JSP Comments

- There are three types of comments permitted in a JSP page:
- **HTML comments**
  - <!-- This is an HTML comment. -->
- **JSP page comments**
  - <%-- This is a JSP comment. It will only be seen in the JSP--%>
- **Java technology comments**
  - <%

    /* This is a Java comment. It will show up in the servlet code. It will not show up in the response. */

    %>

JSP Directives

A JSP directives affects the overall structure of servlet class

- **Directive Types**
  - Page Directive
  - Include Directive
  - Taglib Directive
- **Syntax:**

  <%@ DirectiveName [attr=”value”]* %>
- **Examples:**

  <%@ page …. %> Defines page dependent attribute like import,language,error page etc…

  <%@ include …. %> Includes a file during the translation phase.

  <%@ taglib …%>Declares tag library, contains custom actions.

Page Directives

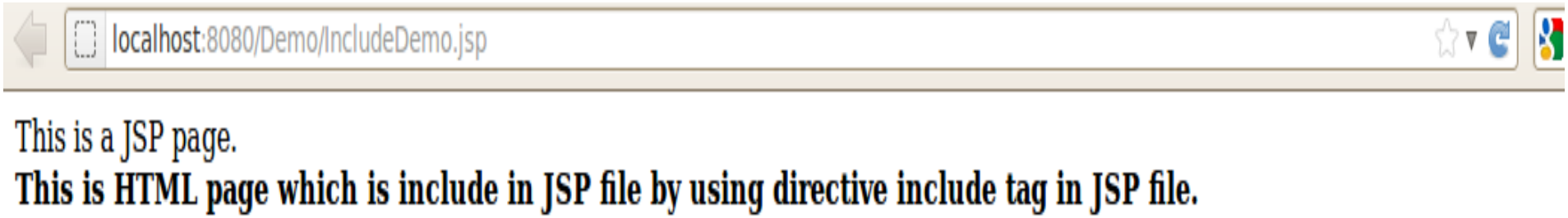
Attribute	Purpose	Syntax	Default Value
buffer	This tells the server about the language to be used in the JSP file. Presently the only valid value for this attribute is java.	buffer="*size*kb|none"	Buffer="8kb"
autoFlush	Controls the behavior of the servlet output buffer.	autoflush="**true**|false"	autoflush="**true**"
contentType	Defines the character encoding scheme.	contentType="*MIME-Type*"	contentType="text/html; charset=ISO-8859-1"
errorPage	Defines the URL of another JSP that reports on Java unchecked runtime exceptions.	errorPage="*url*"	Nothing

Attribute	Purpose	Syntax	Default Value
info	Defines a string that can be accessed with the servlet's getServletInfo() method.	info="*message*"	Nothing
isThreadSafe	Defines the threading model for the generated servlet.	isThreadSafe="**true**|false"	isThreadSafe="**true**"
language	Defines the programming language used in the JSP page.	language="java"	language="java"
session	Specifies whether or not the JSP page participates in HTTP sessions	session="**true**|false"	session="**true**"
isELIgnored	Specifies whether or not EL expression within the JSP page will be ignored.	isELIgnored=" true|**false** "	isELIgnored="**false** "
isScriptingEnabled	Determines if scripting elements are allowed for use.	isScriptingEnabled=“**true**/false”	isScriptingEnabled=“**true**”

Attribute	Purpose	Syntax	Default Value
import	Specifies a list of packages or classes for use in the JSP as the Java import statement does for Java classes.	import="*package.class*"	
Extends	Specifies a superclass that the generated servlet must extend	extends="*package.class* ”	Nothing (Never use)
isErrorPage	Indicates if this JSP page is a URL specified by another JSP page's errorPage attribute.	isErrorPage="true|**false** "	isErrorPage =“**false**"
pageEncoding	This defines data type of page encoding.	pageEncoding="ISO-8859-1"	pageEncoding="ISO-8859-1"

Include Directive

- Include Directive in JSP is an instruction from JSP to JSP Engine about including static file at Transalation Time.
  - Syntax : <%@ include file="fileName" %>
- **file attribute** …. specifies the relative path of the included file.
- This file can be any html, text or jsp file.



Taglib Directive

- The **taglib** directive declares that your JSP page uses a set of custom tags, identifies the location of the library, and provides a means for identifying the custom tags in your JSP page.
  - Syntax: **<%@ taglib uri="uri" prefix="prefixOfTag" >**
- "uri" Attribute: uri attribute in Taglib Directive defines the location of taglib.
- "prefixOfTag" Attribute: prefix Attribute in Taglib Directive defines the prefix that can be used in JSP with custom tag as prefix.

Scriplet

- JSP Declaration(For Defining Variable as well as Method)
  - A declaration declares one or more variables or methods that you can use in Java code later in the JSP file.

    Syntax **<%! declaration; [declaration;]+ ... %>**
- JSP Expression (For Write JAVA Code in JSP page)
  - A expression element contains a scripting language expression that is evaluated.

    Syntax **<%! declaration; [declaration;]+ ... %>**
- JSP Scriplets (For Printing and Write Expression)
  - It contains Java Code.

    Syntax **<% any code %>**

Implicit Object

JSP Implicit Objects are the Java objects that the JSP Container makes available to developers in each page and developer can call them directly without being explicitly declared.
Example :
https://github.com/TopsCode/Java/tree/master/Module-4/4.5Implicit_Object/WebContent

Object	Description
request	This is the **HttpServletRequest** object associated with the request
response	This is the **HttpServletResponse** object associated with the response to the client.
out	This is the **PrintWriter** object used to send output to the client.
session	This is the **HttpSession** object associated with the request.
application	This is the **ServletContext** object associated with application context.
config	This is the **ServletConfig** object associated with the page.
pageContext	This encapsulates use of server-specific features like higher performance **JspWriters**.

Implicit Object continue...

Object	Description
page	This is simply a synonym for **this**, and is used to call the methods defined by the translated servlet class.
Exception	The **Exception** object allows the exception data to be accessed by designated JSP.

Action Tags

You can dynamically insert a file, reuse JavaBeans components, forward the user to another page, or generate HTML for the Java plugin.

Syntax	Purpose
jsp:include	Includes a file at the time the page is requested
jsp:forward	Forwards the requester to a new page
Jsp:useBean	Finds or instantiates a JavaBean
jsp:getProperty	Inserts the property of a JavaBean into the output
jsp:setProperty	Sets the property of a JavaBean
jsp:element	Defines XML elements dynamically.
jsp:attribute	Defines dynamically defined XML element's attribute.
jsp:plugin	Generates browser-specific code that makes an OBJECT or EMBED tag for the Java plugin
jsp:body	Defines dynamically defined XML element's body.
jsp:output	Use to write template text in JSP pages and documents.

JavaBean Component

- A JavaBeans component is a Java class that:
- Has properties defined with accessor and mutator methods (get and set methods)
- Has a no-argument constructor
- Has no public instance variables
- Implements the java.io.Serializable interface
- A JavaBeans component is not a component based on the Enterprise JavaBeansTM specification (EJB component) component

UseBean Tag

- Forms are a very common method of interactions in web sites.The standard way of handling forms in JSP is to define a "bean".
- For that you just need to define a class that has a field corresponding to each field in the form. The class contains the "setters" and “getter” method of the form fields.
- By this action element **<jsp:useBean>,** we use java bean in a JSP page.
- Syntax:

  <jsp:useBean id=*"bean id"* class="bean's class"
  scope="**page**|request|session| application" />

  - id: name of bean
  - scope: location of bean (default is page)
  - class: fully qualified classname
  - scope: session, page, application, request

GetProperty Tag

- This element retrieves the value of a bean property, converts it to a string, and inserts it into the output.
- **Syntax :**

  <jsp:getProperty name="bean's id" property="property name“ param=“name”|value=“name”/>

SetProperty Tag

- This element set the value of a bean property, converts it to a string, and inserts it into the output.
- **Syntax :**

  <jsp:setProperty name="bean's id" property="property name" value="property value"/>

https://github.com/TopsCode/Java/tree/master/Module-4/4.6%20JSP_Action_Tag

Expression Language(EL)

- SP Expression Language (EL) makes it possible to easily access application data stored in JavaBeans components. JSP EL allows you to create expressions both **(a)** arithmetic and **(b)** logical. Within a JSP EL expression, you can use integers, floating point numbers, strings, the built-in constants true and false for boolean values, and null.
- The Expression Language or EL as it is known is used by JSP developers to access and use application data without using java code. EL was introduced in JSTL 1.0, but now is formally defined in JSP 2.0. In this tutorial you will learn the following:
  - Syntax and correct usage of EL
  - EL Expression and Operators
  - Implicit Objects
  - Common mistakes and pitfalls in using EL

Syntax of EL

- Typically, when you specify an attribute value in a JSP tag, you simply use a string. For example:

  <jsp:setProperty name="box"
  property="perimeter" value="100"/>

- JSP EL allows you to specify an expression for any of these attribute values. A simple syntax for JSP EL is as follows:
- **${expr}** Here **expr** specifies the expression itself. The most common operators in JSP EL are **.** and **[]**. These two operators allow you to access various attributes of Java Beans and built-in JSP objects.

JSP EL Implicit Objects

Implicit object	Description
pageScope	Scoped variables from page scope
requestScope	Scoped variables from request scope
sessionScope	Scoped variables from session scope
applicationScope	Scoped variables from application scope
param	Request parameters as strings
paramValues	Request parameters as collections of strings
header	HTTP request headers as strings
headerValues	HTTP request headers as collections of strings
initParam	Context-initialization parameters
cookie	Cookie values
pageContext	The JSP PageContext object for the current page

JSP Standard Tag Library(JSTL)

- The JavaServer Pages Standard Tag Library (JSTL) is a collection of useful JSP tags which encapsulates core functionality common to many JSP applications.
- JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags. It also provides a framework for integrating existing custom tags with JSTL tags.
- The JSTL tags can be classified, according to their functions, into following JSTL tag library groups that can be used when creating a JSP page:
  - **Core Tags**
  - **Formatting tags**
  - **SQL tags**
  - **XML tags**
  - **JSTL Functions**

Core JSTL Tag Library

- **Syntax for defining core tags**

  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

C:CHOOSE/WHEN/OTHERWISE

C:CATCH

C:URL and C:PARAM

C:REDIRECT

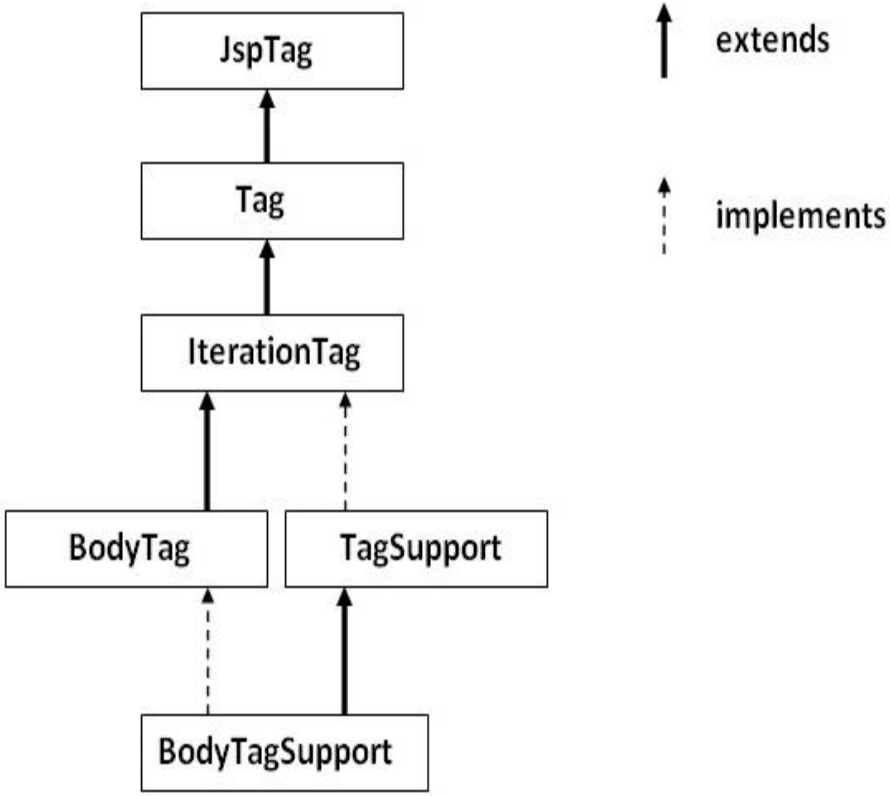
C:OUT

C:IMPORT

C:FOREACH and C:FORTOKENS

C:IF

Custom Tag API



JspTag interface

The JspTag is the root interface for all the interfaces and classes used in custom tag. It is a marker interface.

Tag interface

The Tag interface is the sub interface of JspTag interface. It provides methods to perform action at the start and end of the tag.

TagSupport class

The TagSupport class implements the IterationTag interface. It acts as the base class for new Tag Handlers. It provides some additional methods also.

Syntax to use custom tag

- There are two ways to use the custom tag. They are given below:

<prefix:tagname attr1=value1....attrn=valuen />

<prefix:tagname attr1=value1....attrn=valuen >

    body code

</prefix:tagname>

Example :

https://github.com/TopsCode/Java/tree/master/Module-4/4.8JSTL_Custome_Tag_Practical

Module 5 [Application to Industry]

- Session Management
- Web Service
- MVC Pattern
- AJAX with Web services
- Cookies

Introduction

- A pattern is a proven solution to a problem in a context.
- Design patterns represent a solutions to problems that arise when developing software within a particular context.
- Design patterns can speed up the development process by providing tested, proven development paradigms
- Reusing design patterns helps to prevent subtle issues that can cause major problems, and it also improves code readability for coders and architects who are familiar with the patterns.

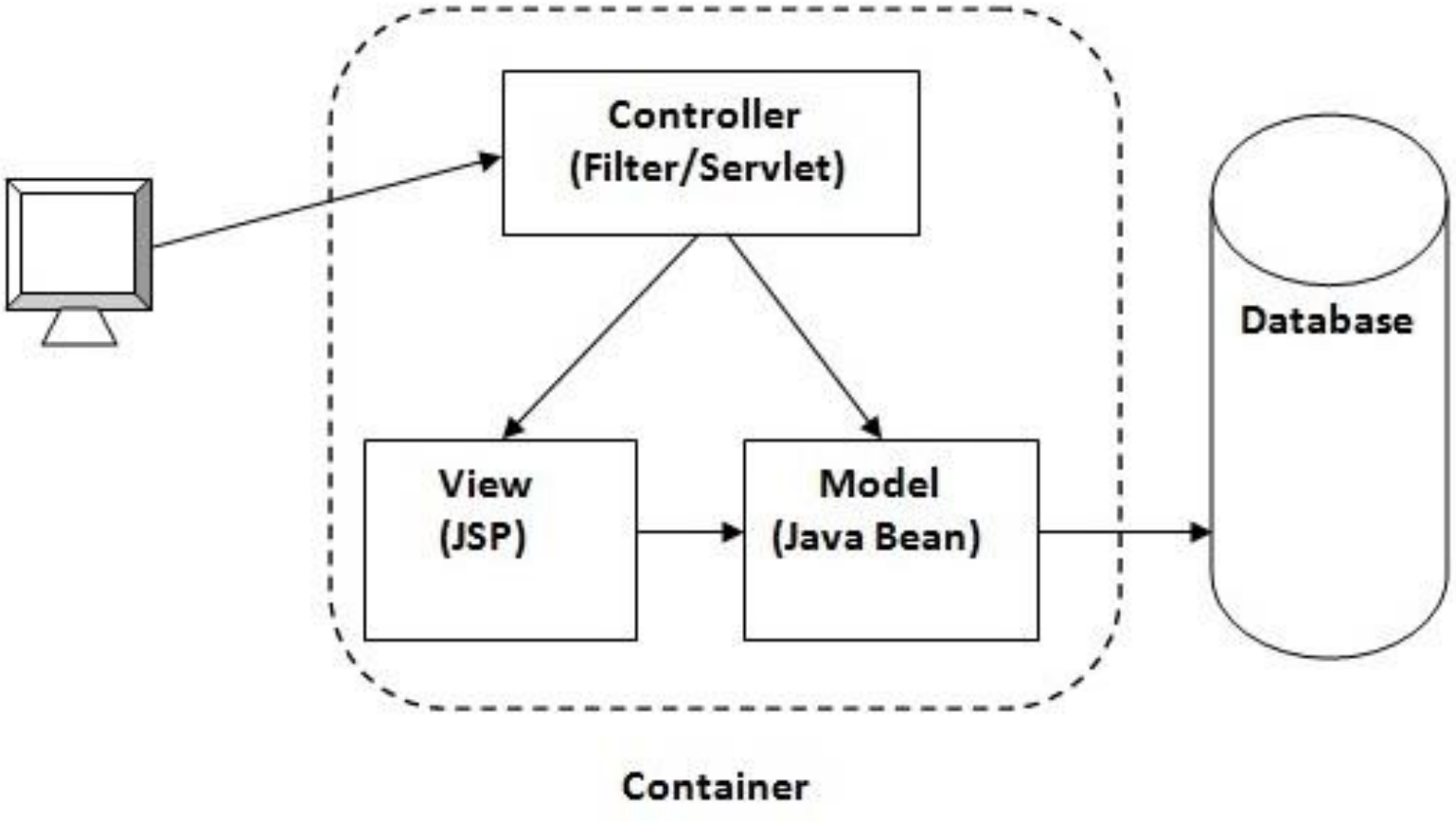
Use of Design Pattern

- In software engineering, a design pattern is a general reusable solution to a commonly occurring problem within a given context in software design.
- It is a description or template for how to solve a problem that can be used in many different situations
- Object-oriented design patterns typically show relationships and interactions between classes or objects, without specifying the final application classes or objects that are involved

- **Name** (essence of the pattern)
  - Model View Controller MVC
- **Context** (where does this problem occur)
  - MVC is an architectural pattern that is used when developing interactive application such as a shopping cart on the Internet.
- **Problem** (definition of the reoccurring difficulty)
  - User interfaces change often, especially on the internet where look-and-feel is a competitive issue. Also, the same information is presented in different ways. The core business logic and data is stable.

- **Solution** (how do you solve the problem)
  - Use the software engineering principle of "separation of concerns" to divide the application into three areas:
    - **Model** encapsulates the core data and functionality
    - **View** encapsulates the presentation of the data there can be many views of the common data
    - **Controller** accepts input from the user and makes request from the model for the data to produce a new view.

MVC Architecture



- **Model** represents the structure of the data in the application, as well as application-specific operations on data (database operations, business logic, processing order)
  - CartModel, InventoryModel, CustomerBean, and others
- **Views** are Java server pages (JSPs)
  - rendered from the web container to the browser, stand-alone applications that provide View functionality, and interfaces to spreadsheet programs, such as the StarOffice suite.
- **Controller** is server side java program (Servlet)
  - MainServlet.java, which dispatches browser requests to other controller objects, such as ShoppingClientController.java, AdminClientController.java, and their related support classes.

Advantages

- We make changes without bringing down the server.
- We leave the core code alone
- We can have multiple versions of the same data displayed
- We can test our changes in the actual environment
- We have achieved “separation of concerns”
- Separating Controller from View (application behavior from presentation)
  - permits run-time selection of appropriate Views based on workflow, user preferences, or Model state.
- Separating Controller from Model (application behavior from data representation)
  - allows configurable mapping of user actions on the Controller to application functions on the Model.

Example :
https://github.com/TopsCode/Java/tree/master/Module-4/4.0.2CrudOperation

Session Management

- Session Management Introduction
- What are the needs?
- Session Management API
- Session Tracking Technique
  - Cookies
  - URL Rewriting
  - Hidden Form Field
  - Session
  - Example for SessionTraking

Introduction

- HTTP protocol and Web Servers are stateless, what it means is that for web server every request is a new request to process and they can't identify if it's coming from client that has been sending request previously.
- But sometimes in web applications, we should know who the client is and process the request accordingly. For example, a shopping cart application should know who is sending the request to add an item and in which cart the item has to be added or who is sending checkout request so that it can charge the amount to correct client.
- **Session** is a conversional state between client and server and it can consists of multiple request and response between client and server. Since HTTP and Web Server both are stateless, the only way to maintain a session is when some unique information about the session (session id) is passed between server and client in every request and response.

HttpSession Object

Methods	Description
public void setAttribute(String name, Object value)	This method binds an object to this session, using the name specified.
public void removeAttribute(String name)	This method removes the object bound with the specified name from this session.

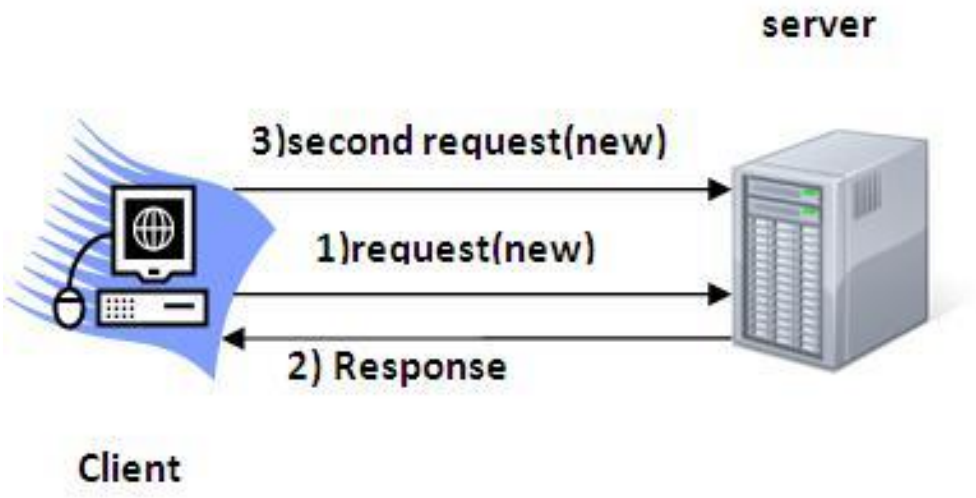
Methods	Description
public void invalidate()	This method invalidates this session and unbinds any objects bound to it.
public long getCreationTime()	This method returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
public String getId()	This method returns a string containing the unique identifier assigned to this session.
public long getLastAccessedTime()	This method returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT.
public int getMaxInactiveInterval()	This method returns the maximum time interval, in

Deleting Session Data

Methods	Description
Remove a particular attribute	You can call *public void removeAttribute(String name)* method to delete the value associated with a particular key.
Delete the whole session	You can call *public void invalidate()* method to discard an entire session.
Setting Session timeout	You can call *public void setMaxInactiveInterval(int interval)* method to set the timeout for a session individually.
Log the user out	The servers that support servlets 2.4, you can call **logout** to log the client out of the Web server and invalidate all sessions belonging to all the users.
web.xml Configuration	If you are using Tomcat, apart from the above mentioned methods, you can configure session time out in web.xml file as follows.

Session Tracking

- **Session** simply means a particular interval of time. Session Tracking is a way to maintain state of an user. Http protocol is a stateless protocol. Each time user requests to the server, server treats the request as the new request. So we need to maintain the state of an user to recognize to particular user.



Session Tracking Techniques

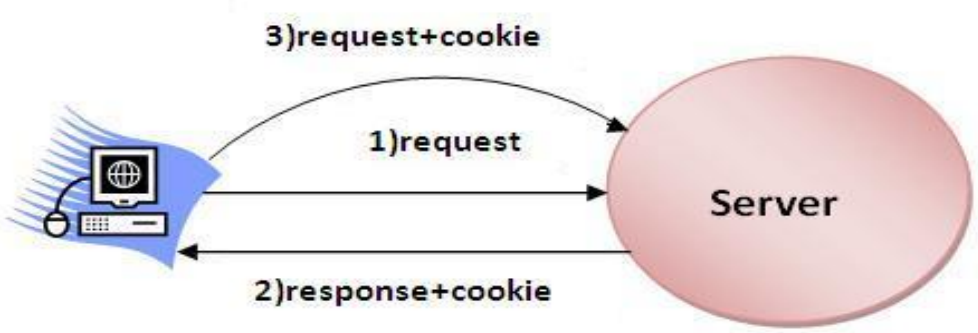
Way of Session Management	Description
Cookies	Cookies are small piece of information that is sent by web server in response header and gets stored in the browser cookies. When client make further request, it adds the cookie to the request header and we can utilize it to keep track of the session. We can maintain a session with cookies but if the client disables the cookies, then it won't work.
URL Rewriting	We can append a session identifier parameter with every request and response to keep track of the session. This is very tedious because we need to keep track of this parameter in every response and make sure it's not clashing with other parameters.

Session Tracking Techniques

Way of Session Management	Description
HTML Hidden Form Field	We can create a unique hidden field in the HTML and when user starts navigating, we can set its value unique to the user and keep track of the session. This method can't be used with links because it needs the form to be submitted every time request is made from client to server with the hidden field. Also it's not secure because we can get the hidden field value from the HTML source and use it to hack the session.
Session (HttpSession Object)	This is the very common way where we user can provide authentication credentials from the login page and then we can pass the authentication information between server and client to maintain the session. This is not very effective method because it wont work if the same user is logged in from different browsers.

Cookie Management

- Cookies are text files stored on the client computer and they are kept for various information tracking purpose. Java Servlets transparently supports HTTP cookies.
- A cookie is a small piece of information that is persisted between the multiple client requests.A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.
- There are three steps involved in identifying returning users:
  - Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
  - Browser stores this information on local machine for future use.
  - When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.



- **Advantage of Cookies**
  - Simplest technique of maintaining the state.
  - Cookies are maintained at client side.
- **Disadvantage of Cookies**
  - It will not work if cookie is disabled from the browser.
  - Only textual information can be set in Cookie object.

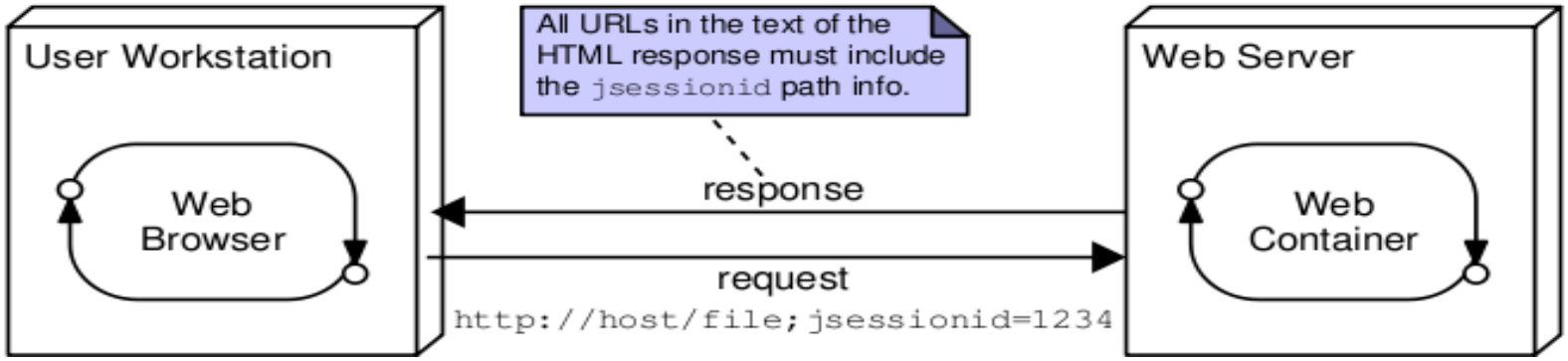
Example :
https://github.com/topscode/java/tree/master/module-5/5.0.1.sessionwithcookies

Cookie Methods

Method	Description
public void setMaxAge(int expiry)	This method sets how much time (in seconds) should elapse before the cookie expires. If you don't set this, the cookie will last only for the current session.
public int getMaxAge()	This method returns the maximum age of the cookie, specified in seconds, By default, -1 indicating the cookie will persist until browser shutdown.
public String getName()	This method returns the name of the cookie. The name cannot be changed after creation.
public void setValue(String newValue)	This method sets the value associated with the cookie.
public String getValue()	This method gets the value associated with the cookie.

URL Rewritting

- You can append some extra data on the end of each URL that identifies the session, and the server can associate that session identifier with data it has stored about that session.
- For example, with http://topsint.com/index.htm;sessionid=12345654560, the session identifier is attached as sessionid=12345654560 which can be accessed at the web server to identify the client.
- URL rewriting is a better way to maintain sessions and works for the browsers when they don't support cookies but here drawback is that you would have generate every URL dynamically to assign a session ID though page is simple static HTML page.



- In URL rewriting, we append a token or identifier to the URL of the next Servlet or the next resource. We can send parameter name/value pairs using the following format:
  - **url?name1=value1&name2=value2&name3=value3**
- A name and a value is separated using an equal = sign, a parameter name/value pair is separated from another parameter using the ampersand(&). When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server. From a Servlet, we can use getParameter() method to obtain a parameter value.
- **Advantage of URL Rewriting**
  - It will always work whether cookie is disabled or not (browser independent).
  - Extra form submission is not required on each pages.
- **Disadvantage of URL Rewriting**
  - It will work only with links.
  - It can send Only textual information.

Example :
https://github.com/TopsCode/Java/tree/master/Module-5/5.0.3.SessionWithUrlReWriting

Hidden Form Field

- A web server can send a hidden HTML form field along with a unique session ID as follows:
  - <input type="hidden" name="sessionid" value="12345">
- This entry means that, when the form is submitted, the specified name and value are automatically included in the GET or POST data. Each time when web browser sends request back, then session_id value can be used to keep the track of different web browsers.
- This could be an effective way of keeping track of the session but clicking on a regular (<A HREF...>) hypertext link does not result in a form submission, so hidden form fields also cannot support general session tracking.

Example :
https://github.com/TopsCode/Java/tree/master/Module-5/5.0.2.SessionWithHiddenFormField

Session(HttpSession Object)

- Apart from the above mentioned three ways, servlet provides HttpSession Interface which provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.
- The servlet container uses this interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user.
- You would get HttpSession object by calling the public method **getSession()** of HttpServletRequest, as below:
  - **HttpSession session = request.getSession();**
- You need to call *request.getSession()* before you send any document content to the client. Here is a summary of the important methods available through HttpSession object:

Example :
https://github.com/topscode/java/tree/master/module-5/5.0sessiondemo

Destributed Technologies in java

- Introduction of distributed technologies in Java
- Use of distributed technologies in Java
- Explanation of various distributed technologies available in Java
  - RMI
  - EJB
  - WebServices
    - SOAP based WebServices
    - Restful WebServices

Introduction

- **Distributed Applications in Java**
  - Introduction to Distributed Computing
  - Java Object Serialization
  - Java Remote Method Invocation (RMI)
- Client-Server:
  - The *client* is the entity accessing the remote resource and the *server* provides access to the resource. Operationally, the client is the *caller* and the server is the *callee*.

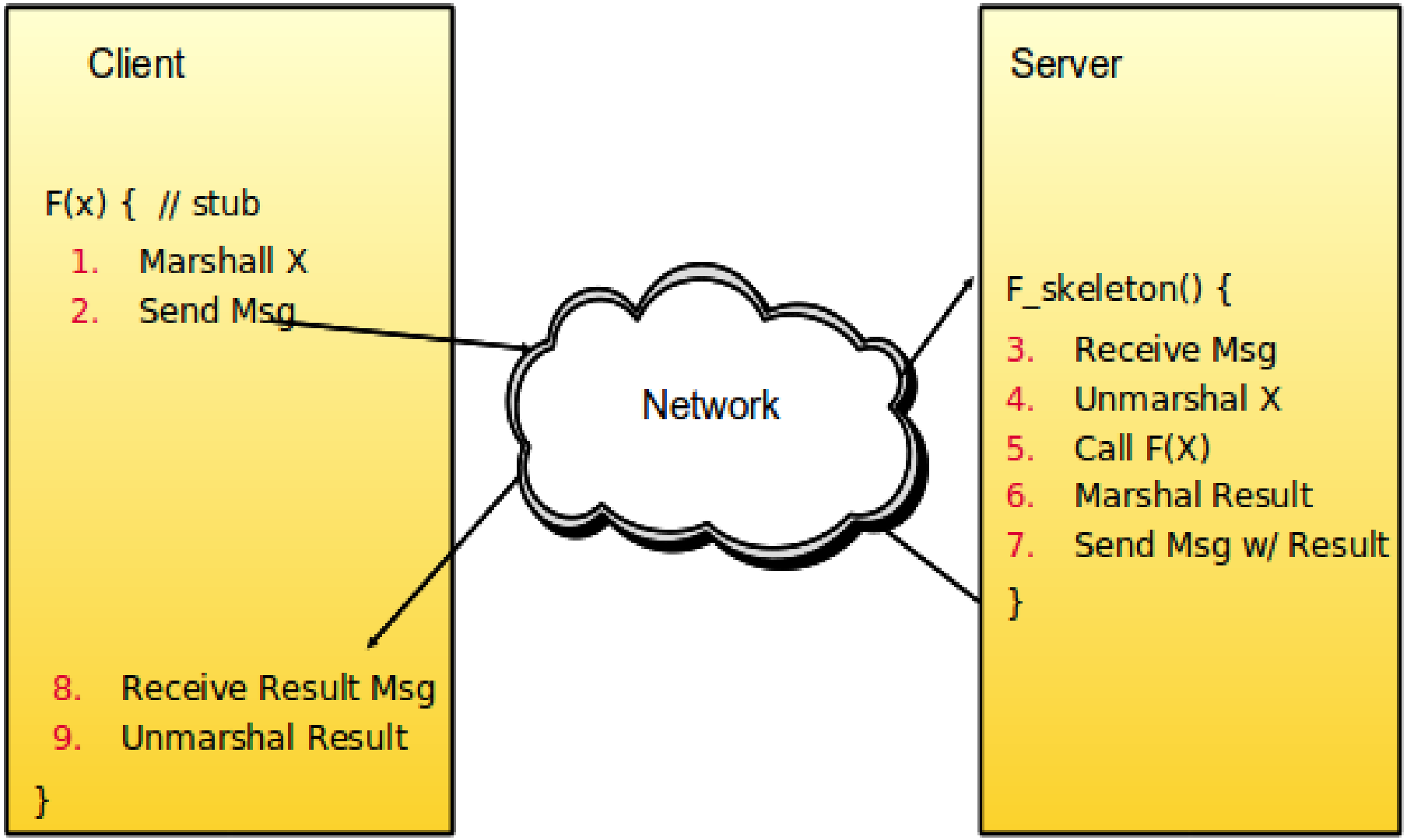
RMI

- In Java terms:
  - The client is the invoker of the method and the server is the object implementing the method.
- The client and the server can be heterogeneous:
  - Different implementation languages
  - Different operating systems
- The roles can be transient
  - The definition is with respect to a particular interaction.
- Client and Server refer both to the code and the system on which the code is running

RMI

- Java RMI allowed programmer to execute remote function class using the same semantics as local functions calls.
- To transfer data from one host to another.
- RMI is a specification that enables one JVM to invoke methods in an object located in another JVM.
- These JVMs could be running on different computers or running as separate pocesses on the same computer
- The server must first bind its name to the registry
- The client lookup the server name in the registry to establish remote references.
- The Stub serializing the parameters to skeleton, the skeleton invoking the remote method and serializing the result back to the stub.

RMI


Client
F(x) { // stub
1. Marshall X
2. Send Msg
8. Receive Result Msg
9. Unmarshal Result
}
Network
Server
F_skeleton() {
3. Receive Msg
4. Unmarshal X
5. Call F(X)
6. Marshal Result
7. Send Msg w/ Result
}

EJB

- **Enterprise JavaBeans** is a specification for creating server-side scalable, transactional, multi-user secure enterprise-level applications. It provides a consistent component architecture framework for creating distributed n-tier middleware. It would be fair to call a bean written to EJB spec a Server Bean.
- A typical EJB Architecture consists of an EJB server, EJB containers that runs on these servers, EJBs that run in these containers, EJB clients and other auxiliary systems like the Java Naming and Directory Interface **(JNDI)** and the Java Transaction Service **(JTS)**.

EJB

- The **Java Naming and Directory Interface (JNDI)** is an application programming interface (API) for accessing different kinds of naming and directory services. 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; distributed objects systems like CORBA, Java RMI, and EJB; and directory services like LDAP, Novell NetWare, and NIS+.
- The **Java Transaction Service (JTS)** is a specification for building a transaction manager that maps onto the Object Management Group (OMG) Object Transaction Service (OTS) used in the Common Object Request Broker Architecture (CORBA) architecture. It uses General Inter-ORB Protocol (IIOP) to propagate the transactions between multiple JTS transaction managers.

What is Web Services?

- Web service is a way of communication that allows interoperability between different applications on different platforms, for example, a java based application on Windows can communicate with a .Net based one on Linux. The communication can be done through a set of XML messages over HTTP protocol.
- Web services are browsers and operating system independent service, which means it can run on any browser without the need of making any changes. Web Services take Web-applications to the Next Level.

Why Web Services?

- Reuse already developed(old) functionality into new software:
- Web Services allow the business logic of many different systems to be exposed over the Web. This gives your applications the freedom to chose the Web Services that they need. Instead of re-inventing the wheel for each client, you need only include additional application-specific business logic on the client-side. This allows you to develop services and/or client-side code using the languages and tools that you want.

Types of Web Services

- SOAP Web Service
  - Simple Object Access Protocol (SOAP) is a standard protocol specification for message exchange based on XML.
  - Communication between the web service and client happens using XML messages. SOAP defines the rules for communication like what are all the tags that should be used in XML and their meaning.
- RESTful Web Service
  - RESTful web service uses architectures that use HTTP or similar protocols by restricting the interface to use standard operations like GET, POST, PUT, DELETE for HTTP.

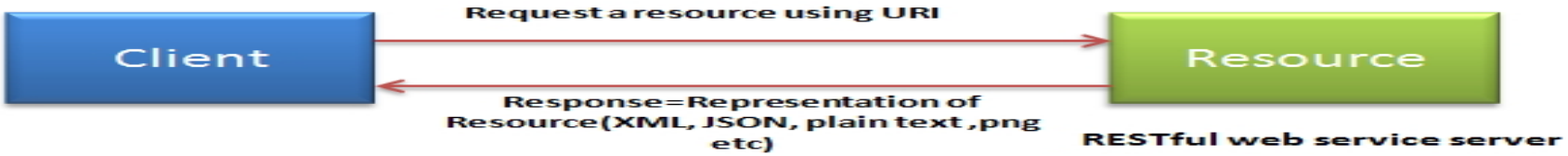
Restful Web Services Introduction

- REST is an architectural style which was brought in by Roy Fielding in 2000 in his doctoral thesis.
- Representational State Transfer (REST) is a stateless client-server architecture in which the web services are viewed as resources and can be identified by their URIs. Web service clients that want to use these resources access via globally defined set of remote methods that describe the action to be performed on the resource.
- It consists of two components REST server which provides access to the resources and a REST client which accesses and modify the REST resources.

Restful WebServices Introduction

- In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol.REST isn't protocol specific, but when people talk about REST they usually mean REST over HTTP.
- The response from server is considered as the representation of the resources. This representation can be generated from one resource or more number of resources.

Restful WebServices Introduction



- RESTful web services use HTTP protocol methods for the operations they perform.Methods are:
- GET:It defines a reading access of the resource without side-effects.This operation is idempotent i.e.they can be applied multiple times without changing the result
- PUT : It creates a new resource.It must also be idempotent.
- DELETE : It removes the resources. The operations are idempotent i.e. they can get repeated without leading to different results.
- POST :It updates an existing resource or creates a new resource.

Restful Web Services Annotations

- **@Path()**
- Its a Class & Method level of annotation
- This will check the path next to the base URL

Syntax :

Base URL : *http://localhost:(port)/<YourApplicationName>/<UrlPattern In Web.xml>/<path>*

Here *<path> is the part of URI*, and this will be identified by *@path* annotation at class/method level, you will be able to understand in the next RESTful hello world tutorial.

Restful Web Services Annotations

- **@GET**

  Its a method level of annotation, this annotation indicates that the following method should respond to the HTTP GET request only, i mean if we annotate our method with @GET, the execution flow will enter that following method if we send GET request from the client

@POST

Its a method level of annotation, this annotation indicates that the following method should respond to the HTTP POST request only

Restful Web Services Annotations

- **@PUT**

  Its a method level of annotation, this annotation indicates that the following method should respond to the HTTP PUT request only.

- **@DELETE**

  Its a method level of annotation, this annotation indicates that the following method should respond to the HTTP DELETE request only.

Restful Web Services Annotations

- **@Produces**
- Its a method or field level annotation, This tells which MIME type is delivered by the method annotated with @GET. I mean when ever we send a HTTP GET request to our RESTful service, it will invokes particular method and produces the output in different formats. There you can specifies in what are all formats (MIME) your method can produce the output, by using @produces annotation. **Remember**: We will use @Produces annotation for GET requests only.

Restful Web Services Annotations

- **@Consumes**

  This is a class and method level annotation, this will define which MIME type is consumed by the particular method. I mean in which format the method can accept the input from the client.

Example :

https://github.com/TopsCode/Java/tree/master/Module-5/5.1Web%20Service%20Demo/RestClient

Example :

https://github.com/TopsCode/Java/tree/master/Module-5/5.1Web%20Service%20Demo/RestServer

RMI

- **Remote Method Invocation** (RMI) is an API which allows an object to invoke a method on an object that exists in another address space, which could be on the same machine or on a remote machine.
- Through RMI, object running in a JVM present on a computer (Client side) can invoke methods on an object present in another JVM (Server side).
- The RMI provides remote communication between the applications using two objects *stub* and *skeleton*.

Working Of RMI

- The communication between client and server is handled by

  using two intermediate objects:

  1. Stub object (on client side)
  2. Skeleton object (on server side).

STUB

- The stub object on the client machine builds an information block and sends this information to the s erver.

The block consists of :

1. An identifier of the remote object to be used
2. Method name which is to be invoked
3. Parameters to the remote JVM

SKELETON

- The skeleton is an object, acts as a gateway for the server side object.
- All the incoming requests are routed through it. When the skeleton receives the incoming request.

It does the following tasks:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshals) the result to the caller.

Create RMI Program Steps

1. Provide the implementation of the remote interface
2. Compile the implementation class and create the stub and skeleton objects using the rmic tool
3. Start the registry service by rmiregistry tool
4. Create and start the remote application
5. Create and start the client application

PayUMoney Integration

- First Go to thia url for a payumoney signup for a merchant user with a proper name and mobile number.
- https://www.payumoney.com/
- After Signup click on a Integration.
- You need to note down Merchant Key and Merchant Salt
- You will also be given Test credentials.
- Then choose how you want to integrate(Java) and proceed
- Open pom.xml file.
- Also open pom.xml file from downloaded kit and copy from <dependencies> to </dependencies> and paste to your projects pom.xml file before <build> tag and save it.
- That will download all the necessary library from the internet.
- Now open index.html and replace key with your payumoney merchant key that you get from payumoney site.

PayUMoney Integration

- Then open JavaIntegrationKit.java and put merchant key and salt key as shown in the screen
- Now create success.jsp in WebApp and just print Hello for the success url.
- Note down the path of success.jsp that we need while running the app.
- Now run index.html, fill necessary fields and in fail and success url part put success.jsp's path for temporary output.
- You will be given testing credentials for education purpose use this to make payment and you will be done.

Example :

https://github.com/TopsCode/Java/tree/master/Module-4/4.10%20Payment%20Integration

Firebase

Firebase evolved from Envolve, a prior startup founded by James Tamplin and Andrew Lee in 2011. Envolve provided developers an API that enables the integration of online chat functionality into their websites. After releasing the chat service, Tamplin and Lee found that it was being used to pass application data that weren't chat messages. Developers were using Envolve to sync application data such as game state in real time across their users. Tamplin and Lee decided to separate the chat system and the real-time architecture that powered it. They founded Firebase as a separate company in April 2012.

Firebase Inc. raised seed funding in May 2012. The company further raised Series A funding in June 2013. In October 2014, Firebase was acquired by Google. In October 2015, Google acquired Divshot to merge it with the Firebase team. Since the acquisition, Firebase has grown inside Google and expanded their services to become a unified platform for mobile developers. Firebase now integrates with various other Google services to offer broader products and scale for developers. In January 2017, Google acquired Fabric and Crashlytics from Twitter to join those services to the Firebase team. Firebase launched Cloud Firestore, a Document Database, in October 2017.

Firebase Crud Operation

- First go on Firebase website. https://firebase.google.com/
- Now go on Firebase Console. https://console.firebase.google.com/u/0/?pli=1
- Now add one project on console.
- Now create one project
- Now click on Developer Tab. In Developer tab click on Database . https://console.firebase.google.com/u/0/project/myproject1-34bf5/overview
- Now you want configuration of Firebase for your project.
- Go to project setting and copy the configuration details.
- Now Create Dynamic Web Project in Eclipse.
- Create one JSP file in Web Content.
- Now add Firebase Configuration details between the Script Tag<script></script>.

Example :

https://github.com/TopsCode/Java/tree/master/Module-4/4.9%20FirebaseCrud/WebContent

•

Module-6 [Hibernate]

- Introduction to Hibernate
- Hibernate Mapping
- Hibernate Query Language
- Relationship

Module-6 [Hibernate]

- Introduction to Hibernate Framework
- Hibernate Architecture
- Hibernate Configuration
- All Core Interfaces
- Query and Criteria
- Named Query

- **All Relationship**
  - One to one
  - One to Many
  - Many to One
  - Many to Many
- All Database operations with hibernate

What is Hibernate Framework

- Hibernate Framework simplifies the development of java application to interact with the database. Hibernate is an Open Source, lightweight, ORM(Object Relational Mapping) tool
- 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.
- ORM tool internally uses the JDBC API to interact with the database.

Advantages of Hibernate Framework

- Fast Performance because of cache is internally used in hibernate framework(Two types of cache First level (mandatory), Second Level) .
- Database Independent query because of HQL(Hibernate Query Language) is object oriented version of SQL
- Automatic table creation.
- Simplifies complex join because of using relation mappings.
- Provides query statistics and database status because of Hibernate supports query cache and provide statistics about query and database status.

Why ORM?

- When we work with an object-oriented systems, there's a mismatch between the object model and the relational database. RDBMSs represent data in a tabular format whereas object-oriented languages, such as Java represent it as an interconnected graph of objects. Consider the following Java Class with proper constructors and associated public function.

What is ORM?

- ORM stands for **O**bject-**R**elational **M**apping (ORM) is a programming technique for **converting data between relational databases and object oriented programming languages** such as Java, C# etc. An ORM system has following advantages over plain JDBC
- Object-relational mapping or ORM is a programming method for **mapping the objects to the relational model where entities/classes are mapped to tables**, instances are mapped to rows and attributes of instances are mapped to columns of table.

Advantages of ORM

- Lets business code access objects rather than DB tables.
- Hides details of SQL queries from OO logic.
- Based on JDBC 'under the hood‘
- No need to deal with the database implementation.
- Entities based on business concepts rather than database structure.
- Transaction management and automatic key generation.
- Fast development of application.

What is Persistence?

- **Persistence** is a process of storing the data to some permanent medium and retrieving it back at any point of time even after the application that had created the data ended.

Rules of creating Persistence Class

- All Java classes that will be persisted need a default constructor.
- All classes should contain an ID in order to allow easy identification of your objects within Hibernate and the database. This property maps to the primary key column of a database table.
- All attributes that will be persisted should be declared private and have getXXX and setXXX methods defined in the JavaBean style.
- A central feature of Hibernate, cache, depends upon the persistent class being either non-final, or the implementation of an interface that declares all public methods.
- All classes that do not extend or implement some specialized classes and interfaces required by the EJB framework.
- All classes are by default Serialized. So no need to implement SerializableInterface to bean class.

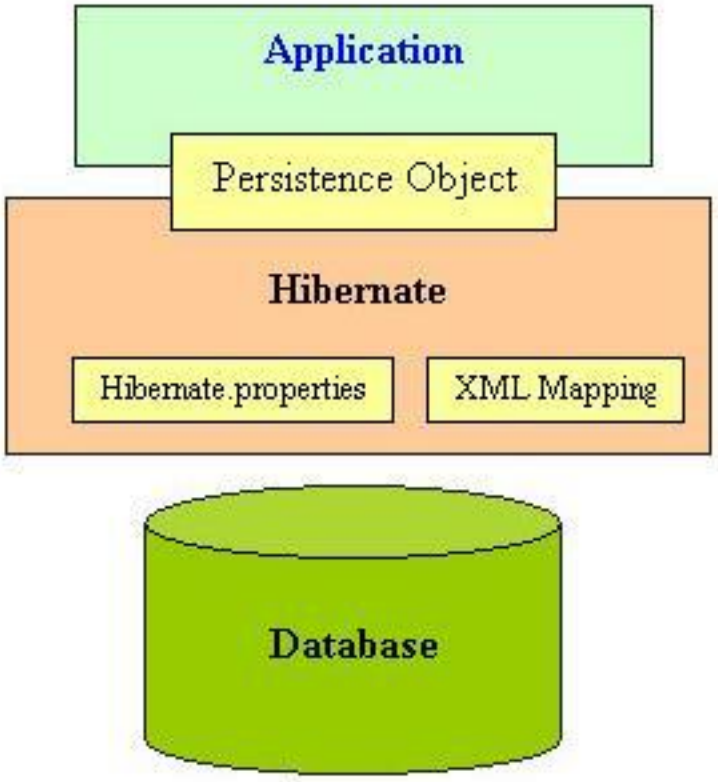
Role of ORM with Persistence Class

- An API to perform basic CRUD operations on objects of persistent classes.
- A language or API to specify queries that refer to classes and properties of classes.
- A configurable facility for specifying mapping metadata.
- A technique to interact with transactional objects to perform dirty checking, lazy association fetching, and other optimization functions.
- There are several persistent frameworks and ORM options in Java. A persistent framework is an ORM service that stores and retrieves objects into a relational database.

Hibernate Architecture

- 4 – levels of hibernate Architecture
- High Level of Hibernate Architecture
- Elements of Hibernate Architecture

Levels of Hibernate Architecture



Explanation of Diagram

- Diagram shows that Hibernate is using the database and configuration data to provide persistence services (and persistent objects) to the application.
- To use Hibernate, it is required to create Java classes that represents the table in the database and then map the instance variable in the class with the columns in the database. Then Hibernate can be used to perform operations on the database like select, insert, update and delete the records in the table. Hibernate automatically creates the query to perform these operations.

Main Components of Hibernate Architecture

- **Connection Management**
  - service provide efficient management of the database connections. Database connection is the most expensive part of interacting with the database as it requires a lot of resources of open and close the database connection.
- **Transaction Management**
  - service provide the ability to the user to execute more than one database statements at a time.
- **Object Relational Mapping**
  - is technique of mapping the data representation from an object model to a relational data model. This part of the hibernate is used to select, insert, update and delete the records form the underlying table. When we pass an object to a **Session.save()** method, Hibernate reads the state of the variables of that object and executes the necessary query.

Elements of Hibernate Architecture

- SessionFactory(org.hibernate.SessionFactory)
- Configuration(org.hibernate.cfg.Configuration)
- Session(org.hibernate.Session)
- Transaction(org.hibernate.Transaction)
- Query(org.hibernate.Query)
- Criteria(org.hibernate.Criteria)
- Type(org.hibernate.Type)
- Generator(org.hibernate.id.IdentifierGeneratar)

Note

- It is called "**Lite**" architecture when we only uses the object relational mapping component.
- While in "**Full Cream**" architecture all the three component Object Relational mapping, Connection Management and Transaction Management) are used.

SessionFactory

- The SessionFactory is a factory of session and client of ConnectionProvider.
- It is factory of JDBC connections. It abstracts the application from DriverManager or DataSource.
- It holds second level cache(optional) of data.
- The org.hibernate.SessionFactory interface provides factory method to get the object of Session.

Configuration

- An instance of Configuration allows the application to specify properties and mapping documents to be used when creating a SessionFactory.
- Usually an application will create a single Configuration, build a single instance of SessionFactory and then instantiate Sessions in threads servicing client requests.
- The Configuration is meant only as an initialization-time object. SessionFactory’s are immutable and do not retain any association back to the Configuration.
- A new Configuration will use the properties specified in hibernate.properties by default.

Session

- The session object provides an interface between the application and data stored in the database.
- It is a short-lived object and wraps the JDBC connection.
- It is factory of Transaction, Query and Criteria. It holds a first-level cache (mandatory) of data.
- The org.hibernate.Session interface provides method to insert(save), update and delete the object.
- It also provides factory method for transaction, Query and Criteria.

Transaction

- The transaction object specifies the atomic unit of work. It is optional.
- The org.hibernate.Transaction interface provides methods for transaction management.

Query

- An object-oriented representation of a Hibernate query.
- A Query instance is obtained by calling **Session.createQuery().**
- This interface exposes some extra functionality beyond that provided by **Session.iterate() and Session.find()**
- a particular page of the result set may be selected by calling setMaxResults(), setFirstResult()
- named query parameters may be used
- the results may be returned as an instance of ScrollableResults
- Queries are executed by calling **list(), scroll() or iterate().** A query may be re-executed by subsequent invocations.
- Its lifespan is, however, bounded by the lifespan of the Session that created it.

Criteria

- Criteria is a simplified API for retrieving entities by composing Criterion objects.
- Hibernate provides alternate ways of manipulating objects and in turn data available in RDBMS tables. One of the methods is Criteria API which allows you to build up a criteria query object programmatically where you can apply filtration rules and logical conditions.
- The Hibernate **Session** interface provides **createCriteria()** method which can be used to create a **Criteria** object that returns instances of the persistence object's class when your application executes a criteria query.

Type

Mapping type	Java type	ANSI SQL Type
integer	int or java.lang.Integer	INTEGER
long	long or java.lang.Long	BIGINT
short	short or java.lang.Short	SMALLINT
float	float or java.lang.Float	FLOAT
double	double or java.lang.Double	DOUBLE
big_decimal	java.math.BigDecimal	NUMERIC
character	java.lang.String	CHAR(1)
string	java.lang.String	VARCHAR
text	java.lang.String	CLOB
byte	byte or java.lang.Byte	TINYINT

Mapping type	Java type	ANSI SQL Type
boolean	boolean or java.lang.Boolean	BIT
yes/no	boolean or java.lang.Boolean	CHAR(1) ('Y' or 'N')
true/false	boolean or java.lang.Boolean	CHAR(1) ('T' or 'F')
date	java.util.Date or java.sql.Date	DATE
time	java.util.Date or java.sql.Time	TIME
timestamp	java.util.Date or java.sql.Timestamp	TIMESTAMP
calendar	java.util.Calendar	TIMESTAMP
calendar_date	java.util.Calendar	DATE

Hibernate Configuration

- Hibernate requires to know in advance where to find the mapping information that defines how your Java classes relate to the database tables. Hibernate also requires a set of configuration settings related to database and other related parameters. All such information is usually supplied as a standard Java properties file called **hibernate.properties**, or as an XML file named **hibernate.cfg.xml**.
- consider XML formatted file **hibernate.cfg.xml** to specify required Hibernate properties in my examples. Most of the properties take their default values and it is not required to specify them in the property file unless it is really required. This file is kept in the root directory of your application's classpath.

Hibernate Properties

Hibernate Properties	Description
hibernate.dialect	This property makes Hibernate generate the appropriate SQL for the chosen database.
hibernate.connection.driver_class	The JDBC driver class.
hibernate.connection.url	The JDBC URL to the database instance.
hibernate.connection.username	The database username.
hibernate.connection.password	The database password.
hibernate.hbm2ddl.auto	It validate | update | create | create-drop the tables.
Hibernate.show_sql	Write all SQL statements to console.
hibernate.connection.pool_size	Limits the number of connections waiting in the Hibernate database connection pool.

Hibernate Properties with MySQL

Hibernate Properties	Description
hibernate.dialect	org.hibernate.dialect.MySQLDialect
hibernate.connection.driver_class	com.mysql.jdbc.Driver
hibernate.connection.url	jdbc:mysql://localhost:3306/hibernate
hibernate.connection.username	root
hibernate.connection.password	root | (nothing)
hibernate.hbm2ddl.auto	validate | update | create | create-drop
Hibernate.show_sql	true | false
hibernate.connection.pool_size	5, 10, 20, 40 (Number format)

Generator

- <generator /> is one of main element we are using in the hibernate framework [in the mapping file], let us see the concept behind this generators.
- Up to now in our hibernate mapping file, we used to write <generator /> in the id element scope, actually this is default like whether you write this assigned generator or not hibernate will takes automatically
- In fact this assigned means hibernate will understand that, while saving any object hibernate is not responsible to create any primary key value for the current inserting object, user has to take the response.
- The thing is, while saving an object into the database, the generator informs to the hibernate that, how the primary key value for the new record is going to generate
- hibernate using different primary key generator algorithms, for each algorithm internally a class is created by hibernate for its implementation.
- hibernate provided different primary key generator classes and all these classes are implemented from **org.hibernate.id.IdentifierGeneratar** Interface
- while configuring <generator /> element in mapping file, we need to pass parameters if that generator class need any parameters, actually one sub element of <generator /> element is <param />, will talk more about this

Example of :

```
<generator class="">
<param name=""> value </param>
</generator>
```

List of Generator Classes

- Assigned - default generator
- Increment - assign primary key and increment by default
- Sequence - create data as sequentially in database
- identity – support to id column
- hilo – high and low algorithm
- native – uses identity, sequence, hilo
- foregin – associated the object to the id
- uuid.hex – 128-bit UUID algorithm in hexadecimal
- uuid.string – 128-bit UUID algorithm in string

Example :

https://github.com/TopsCode/Java/tree/master/Module-6/6.0%20HibernateApp

Object Relation Mapping

- One - to - One Mapping
- One - to Many Mapping
- Many - to - One Mapping
- Many - to - One Mapping

Create OR Mapping by using Collection Mapping

Collection Mapping

If an entity or class has collection of values for a particular variable, then we can map those values using any one of the collection interfaces available in java. Hibernate can persist instances of **java.util.Map, java.util.Set, java.util.SortedMap, java.util.SortedSet, java.util.List**, and any **array** of persistent entities or values.

Classes	Description
Java.util.Set Java.util.SortedSet	This is mapped with a <set> element and initialized with java.util.HashSet
Java.util.List	This is mapped with a <list> element and initialized with java.util.ArrayList
Java.util.Collection	This is mapped with a <bag> or <ibag> element and initialized with java.util.ArrayList
Java.util.Map Java.util.SortedMap	This is mapped with a <map> element and initialized with java.util.HashMap

Class Tag property

<class name="ClassName" table="tableName" />

Property	Description
name (optional)	the name of the property.
table (optional - defaults to the unqualified class name)	the name of its database table.

Id Tag property

```
<id name="propertyName" type="typename" column="column_name" >
<generator class="generatorClass"/>
</id>
```

Property	Description
name (optional)	the name of the identifier property.
type (optional)	a name that indicates the Hibernate type.
column (optional - defaults to the property name)	the name of the primary key column.
generator	<generator> child element names a Java class used to generate unique identifiers for instances of the persistent class.

One-to-One Tag property

```
<one-to-one name="propertyName" class="ClassName" cascade="cascade_style"
    constrained="true|false" fetch="join|select" property-
    ref="propertyNameFromAssociatedClass" access="field|property|ClassName" formula="any
    SQL expression" lazy="proxy|no-proxy|false" entity-name="EntityName“ />
```

Property	Description
name	the name of the property.
class (optional - defaults to the property type determined by reflection)	the name of the associated class.
cascade (optional)	specifies which operations should be cascaded from the parent object to the associated object.

Property	Description
constrained (optional)	specifies that a foreign key constraint on the primary key of the mapped table and references the table of the associated class. This option affects the order in which save() and delete() are cascaded, and determines whether the association can be proxied. It is also used by the schema export tool.
fetch (optional - defaults to select)	chooses between outer-join fetching or sequential select fetching.
property-ref (optional)	the name of a property of the associated class that is joined to the primary key of this class. If not specified, the primary key of the associated class is used.
access (optional - defaults to property)	the strategy Hibernate uses for accessing the property value.
formula (optional)	almost all one-to-one associations map to the primary key of the owning entity. If this is not the case, you can specify another column, columns or expression to join on using an SQL formula. See org.hibernate.test.onetooneformula for an example.

Many-to-One Tag property

<many-to-one name="propertyName" column="column_name" class="ClassName" cascade="cascade_style" fetch="join|select" update="true|false" insert="true|false" property-ref="propertyNameFromAssociatedClass" access="field|property|ClassName" unique="true|false" not-null="true|false" optimistic-lock="true|false" lazy="proxy|no-proxy|false" not-found="ignore|exception" entity-name="EntityName" formula="arbitrary SQL expression" />

Property	Description
name	the name of the property.
column(optional)	the name of the foreign key column. This can also be specified by nested <column> element(s).
class(optional - defaults to the property type determined by reflection)	the name of the associated class.
cascade(optional)	specifies which operations should be cascaded from the parent object to the associated object.

Property	Description
fetch(optional - defaults to select)	chooses between outer-join fetching or sequential select fetching.
update, insert (optional - defaults to true)	specifies that the mapped columns should be included in SQL UPDATE and/or INSERT statements. Setting both to false allows a pure "derived" association whose value is initialized from another property that maps to the same column(s), or by a trigger or other application.
property-ref (optional)	the name of a property of the associated class that is joined to this foreign key. If not specified, the primary key of the associated class is used.
access (optional - defaults to property)	the strategy Hibernate uses for accessing the property value.
unique (optional)	enables the DDL generation of a unique constraint for the foreign-key column. By allowing this to be the target of a property-ref, you can make the association multiplicity one-to-one.

Property	Description
not-null (optional)	enables the DDL generation of a nullability constraint for the foreign key columns.
ptimistic-lock (optional - defaults to true)	specifies that updates to this property do or do not require acquisition of the optimistic lock. In other words, it determines if a version increment should occur when this property is dirty.
lazy (optional - defaults to proxy)	by default, single point associations are proxied. lazy="no-proxy" specifies that the property should be fetched lazily when the instance variable is first accessed. This requires build-time bytecode instrumentation. lazy="false" specifies that the association will always be eagerly fetched.
not-found (optional - defaults to exception)	specifies how foreign keys that reference missing rows will be handled. ignore will treat a missing row as a null association.
entity-name (optional)	the entity name of the associated class.
formula (optional)	an SQL expression that defines the value for a *computed* foreign key.

Many-to-Many Tag property

<many-to-many column="column_name" formula="any SQL expression" class="ClassName" fetch="select|join" unique="true|false" not-found="ignore|exception" entity-name="EntityName" property-ref="propertyNameFromAssociatedClass" />

Property	Description
column (optional)	the name of the element foreign key column.
formula (optional)	an SQL formula used to evaluate the element foreign key value.
class (required)	the name of the associated class.
fetch (optional - defaults to join)	enables outer-join or sequential select fetching for this association. This is a special case; for full eager fetching in a single SELECT of an entity and its many-to-many relationships to other entities, you would enable join fetching,not only of the collection itself, but also with this attribute on the <many-to-many> nested element.

Property	Description
unique (optional)	enables the DDL generation of a unique constraint for the foreign-key column. This makes the association multiplicity effectively one-to-many.
not-found (optional - defaults to exception)	specifies how foreign keys that reference missing rows will be handled: ignore will treat a missing row as a null association.
entity-name (optional)	the entity name of the associated class, as an alternative to class.
property-ref (optional)	the name of a property of the associated class that is joined to this foreign key. If not specified, the primary key of the associated class is used.

Example :
https://github.com/TopsCode/Java/tree/master/Module-6/6.1%20OneToOneHibernateApp

Example :
https://github.com/TopsCode/Java/tree/master/Module-6/6.2%20OneToManyToOneApp

Example :

https://github.com/TopsCode/Java/tree/master/Module-6/6.2%20OneToManyToOneApp

Example :

https://github.com/TopsCode/Java/tree/master/Module-6/6.3%20ManyToManyApp

FROM Clause using HQL

```
String hql = "FROM Employee";
Query query = session.createQuery(hql);
List results = query.list();
```

FROM Clause using Criteria

```
Criteria criteria = session.createCriteria(Employee.class);
List results = criteria .list();
```

Where Clause using HQL

```
String hql = "FROM Employee E WHERE E. salary = 2000 ";
Query query = session.createQuery(hql);
List results = query.list();
```

Restrictions Clause using Criteria

```
Criteria criteria = session.createCriteria(Employee.class);
criteria add(Restrictions.eq("salary", 2000));
List results = criteria .list();
```

Where and AS Clause using HQL

```
String hql = "FROM Employee E WHERE E. salary = 2000 ";
Query query = session.createQuery(hql);
List results = query.list();
```

Restrictions Clause using Criteria

```
Criteria criteria = session.createCriteria(Employee.class);
criteria add(Restrictions.eq("salary", 2000));
List results = criteria .list();
```

OrderBy Clause using HQL

```
String hql = "FROM Employee E WHERE E.id > 10 ORDER BY E.salary
    DESC";
Query query = session.createQuery(hql);
List results = query.list();
```

Sorting using Criteria

```
Criteria criteria = session.createCriteria(Employee.class);
criteria add(Restrictions.gt(“id", 10));
criteria.addOrder(Order.desc("salary"));
List results = criteria .list();
```

Example :
https://github.com/TopsCode/Java/tree/master/Module-6/6.0%20HibernateApp

Module- 7 [Spring Framework]

- Introduction to Spring
- Spring Framework
- Spring IOC Container
- Database Operation with Spring
- Spring ORM
- Spring AOP

Spring Framework Modules

- Spring Core
- Spring MVC

Spring Core

- Overview on Spring Frameworks
- Introduction of Spring Framework Architecture
- Spring IOC Containers
- Spring Hello World using IDE
- Spring Bean Definition
- Spring Dependency Injections
- Spring Aspect Oriented Programming
- Spring JDBC Template
- Spring ORM

Introduction to Spring

- Spring framework is an open source Java platform that provides comprehensive infrastructure support for developing robust Java applications very easily and very rapidly.
- It was developed by Rod Johnson in 2003.
- Spring framework makes the easy development of JavaEE application.
- Spring is a lightweight framework.
- It provides support to various frameworks such as Struts, Hibernate, EJB, JSF etc

Introduction to IOC & DI

- The Spring container is at the core of the Spring Framework.
- The container will create the objects, wire them together, configure them, and manage their complete lifecycle from creation till destruction.
- The Spring container uses dependency injection (DI) to manage the components that make up an application.
- Dependency injection is a software design pattern that allows the removal of hard-coded dependencies and makes it possible to change them, whether at run-time or compile-time.
- Dependency injection is quite simple to use and it has quite a “shallow” learning curve.

Introduction to IOC & DI

- Dependency Injection helps in gluing these classes together and same time keeping them independent.
- Dependency injection can happen in the way of passing parameters to the constructor or by post-construction using setter methods.
- As Dependency Injection is the heart of Spring Framework.
- These are the design patterns that are used to remove dependency from the programming code. They make the code easier to test and maintain.

Introduction to IOC & DI

- Let's understand this with the following code:

```
class Employee{
    Address address;
        Employee(){
        address=new Address();
    }
}
```

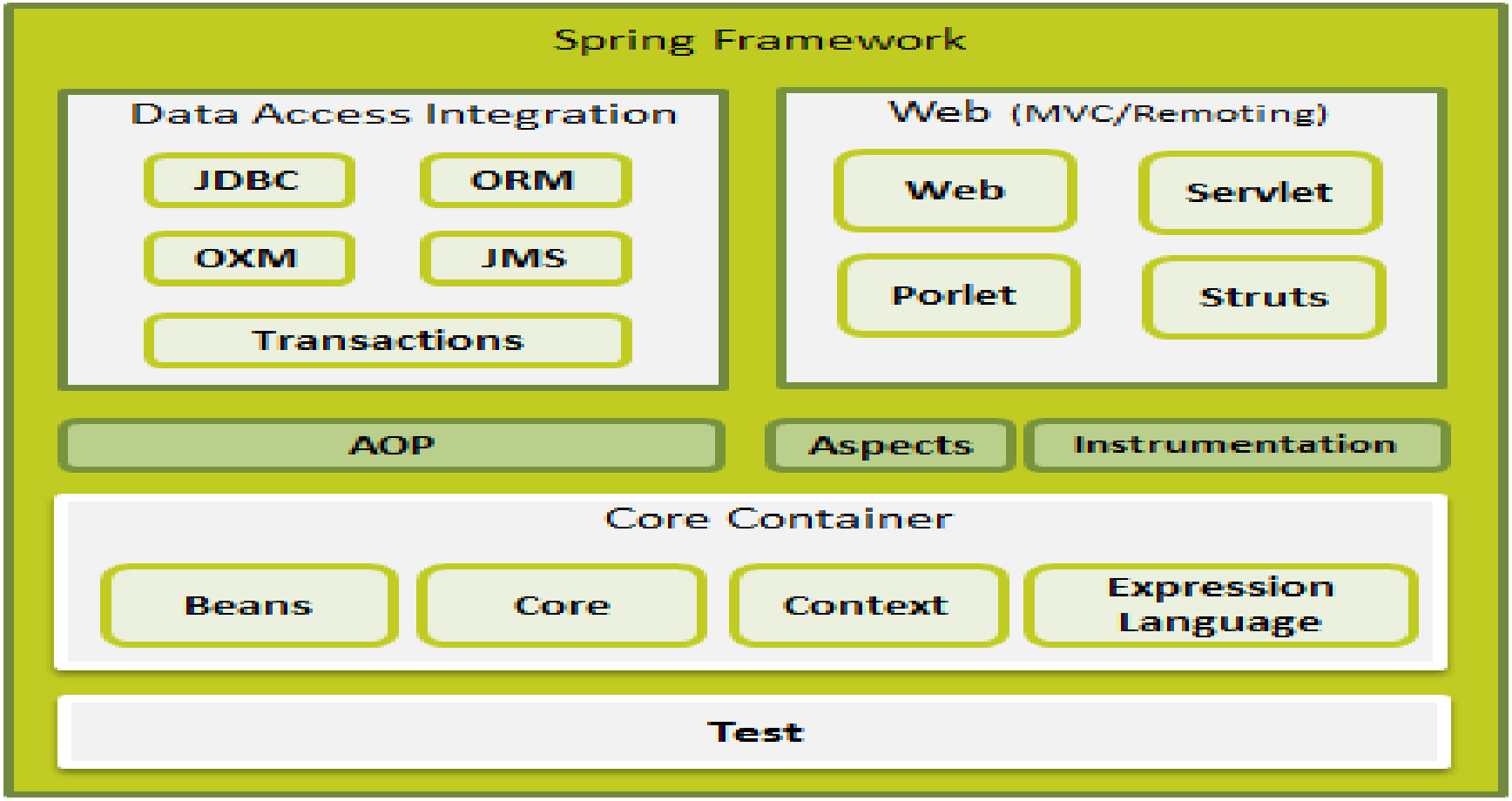
- In such case, there is dependency between the Employee and Address (tight coupling). In the Inversion of Control scenario, we do this something like this:

Introduction to IOC & DI

```
class Employee{
    Address address;
        Employee(Address address){
        this.address=address;
    }
}
```

- Thus, IOC makes the code loosely coupled. In such case, there is no need to modify the code if our logic is moved to new environment.
- In Spring framework, IOC container is responsible to inject the dependency. We provide metadata to the IOC container either by XML file or annotation.

Spring Architecture



Spring Core Container

Core	These modules provide IOC and Dependency Injection features.
Beans	These modules provides BeanFactory and ApplicationContext, which is sophisticated implementation of the factory pattern
Context	module supports internationalization (I18N), EJB, JMS, Basic Remoting.
Expression Language	This module provides a powerful expression language for querying and manipulating an object at runtime.

Data Access/Integration

- This group comprises of JDBC, ORM, OXM, JMS and Transaction modules. These modules basically provide support to interact with the database.

Data Access/Integration	
JDBC	This module provides a JDBC-abstraction layer that removes the need to do tedious JDBC related coding.
ORM	This module provides integration layers for popular object-relational mapping APIs, including JPA, JDO, Hibernate, and iBatis.
OXM	This module provides an abstraction layer that supports Object/XML mapping implementations for JAXB, Castor, XMLBeans, JiBX and XStream.

Data Access/Integration

Data Access/Integration	
JMS(Java Messaging Service):	This module contains features for producing and consuming messages.
Transaction:	module supports programmatic and declarative transaction management for classes that implement special interfaces and for all your POJOs.

Spring – Web Modules

Web	
Web	**:** This module provides basic web-oriented integration features such as multipart file-upload functionality and the initialization of the IoC container using servlet listeners and a web-oriented application context.
Web-Servlet	: This module contains Spring's model-view-controller (MVC) implementation for web applications.
Web-Struts:	: This module contains the support classes for integrating a classic Struts web tier within a Spring application.
Web-Portlet	: This module provides the MVC implementation to be used in a portlet environment and mirrors the functionality of Web-Servlet module.

Spring IOC Containers

- The Spring container is at the core of the Spring Framework.
- The container will create the objects, wire them together, configure them, and manage their complete lifecycle from creation till destruction.
- The Spring container uses dependency injection (DI) to manage the components that make up an application.
- The main tasks performed by IoC container are:
  - to instantiate the application class
  - to configure the object
  - to assemble the dependencies between the objects.

Spring IOC Containers

- The configuration metadata can be represented either by
  - XML,
  - Java annotations, or
  - Java code.
- There are two types of IoC containers. They are:
  - BeanFactory
  - ApplicationContext

Spring IOC Containers

- The
  - org.springframework.beans.factory.xml.XmlBeanFactory ,
  - org.springframework.context.support.ClassPathXmlApplicationContextacts and the
  - org.springframework.context.support.FileSystemXmlApplicationContext as the IoC container.
- The ApplicationContext interface is built on top of the BeanFactory interface.
- It adds some extra functionality than BeanFactory such as simple integration with
  - spring's AOP,
  - message resource handling (for I18N),
  - event propagation,
  - application layer specific context (e.g. WebApplicationContext) for web application.
- So it is better to use ApplicationContext than BeanFactory

IOC - BeanFactory

- The XmlBeanFactory is the implementation class for the BeanFactory interface. To use the BeanFactory, we need to create the instance of XmlBeanFactory class, which is deprecated form the spring 3.0 jar files as given below:

```
~~XmlBeanFactory~~ factory = new XmlBeanFactory(new
    ClassPathResource("Beans.xml"));
```

- The constructor of XmlBeanFactory class receives the Resource object so we need to pass the resource object to create the object of BeanFactory

IOC - ApplicationContext

- The ClassPathXmlApplicationContext class is the implementation class of ApplicationContext interface. We need to instantiate the ClassPathXmlApplicationContext class to use the ApplicationContext as given below:

  ApplicationContext context = new ClassPathXmlApplicationContext("Beans.xml");

  ----------Or----------

  ApplicationContext context = new FileSystemXmlApplicationContext ("D:/TestSpring/Workspace/Spring/HelloWorld/src/Beans.xml");

- The constructor of ClassPathXmlApplicationContext or FileSystemXmlApplicationContext class receives string, so we can pass the name of the xml file to create the instance of ApplicationContext.

Spring Hello World using IDE

- Here, we are going to create a simple application of spring framework using eclipse IDE.
- Let's see the simple steps to create the spring application in Eclipse IDE.
  - Create the java project
  - Add or configure spring jar files
  - Create the class
  - Create the xml file to provide the values
  - Create the main class
  - **Example :**
  - https://github.com/TopsCode/Java/tree/master/Module-7/7.0HelloSpring

Spring Bean Definition

- The objects that form the backbone of your application and that are managed by the Spring IoC container are called beans.
- A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container.
- These beans are created with the configuration metadata that you supply to the container, for example, in the form of XML <bean/> definitions.
- The bean definition contains the information called configuration metadata which is needed for the container to know the followings:

Spring Bean Definition

- How to create a bean
- Bean's lifecycle details
- Bean's dependencies

- All the above configuration metadata translates into a set of the following properties that make up each bean definition.

Spring Bean Definition

- Spring Configuration Metadata
- Spring IoC container is totally decoupled from the format in which this configuration metadata is actually written.
- There are following three important methods to provide configuration metadata to the Spring Container:
  - XML based configuration file.
  - Annotation-based configuration
  - Java-based configuration

Spring Life Cycle

- When bean is initialized it might require to perform some activity before it can come into use able state (State in which application can use it) and when bean is getting destroyed there might be some cleanup activity required for given bean.
- These activities are known as bean Lifecycle.
- Container will contain beans as long as they are required by Application.
- Beans created outside Spring container can also be registered with AC(Application Context).
- BeanFactory is root interface for accessing the bean container.
- Other interfaces are also available for specific purpose.
- BeanFactory is a central registry of application components (Beans).
- These component (Beans) have lifecycle interfaces and methods which will be invoked in some order before Bean can be handed over to application and before Bean is getting destroyed.

Spring Life Cycle

- Through, there are lists of the activities that take place behind the scenes between the time of bean Instantiation and its destruction. To define setup and teardown for a bean, we simply declare the <bean> with init-method and/or destroy-method parameters.
- The init-method attribute specifies a method that is to be called on the bean immediately upon instantiation.
- Similarly, destroy-method specifies a method that is called just before a bean is removed from the container.

Example :
https://github.com/TopsCode/Java/tree/master/Module-7/7.3SpringLifeCycle

Inheritance

- A bean definition can contain a lot of configuration information, including constructor arguments, property values, and container-specific information such as initialization method, static factory method name, and so on.
- A child bean definition inherits configuration data from a parent definition.
- The child definition can override some values, or add others, as needed.
- Spring Bean definition inheritance has nothing to do with Java class inheritance but inheritance concept is same.
- You can define a parent bean definition as a template and other child beans can inherit required configuration from the parent bean.
- When you use XML-based configuration metadata, you indicate a child bean definition by using theparent attribute, specifying the parent bean as the value of this attribute.

Example :
https://github.com/TopsCode/Java/tree/master/Module-7/7.1SpringInheritance

Bean Scope

- To force Spring to produce a new bean instance each time one is needed, you should declare the bean's scope attribute to be prototype.
- Similar way if you want Spring to return the same bean instance each time one is needed, you should declare the bean's scope attribute to be singleton.
- The Spring Framework supports following five scopes, three of which are available only if you use a web-aware ApplicationContext.

Bean Scope

Scope	Description
singleton	This scopes the bean definition to a single instance per Spring IoC container (default).
prototype	This scopes a single bean definition to have any number of object instances.
request	This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext.
session	This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.
global-session	This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

Singleton Scope

- If scope is set to singleton, the Spring IoC container creates exactly one instance of the object defined by that bean definition.
- This single instance is stored in a cache of such singleton beans, and all subsequent requests and references for that named bean return the cached object.

Prototype Scope

- If scope is set to prototype, the Spring IoC container creates new bean instance of the object every time a request for that specific bean is made.
- As a rule, use the prototype scope for all state-full beans and the singleton scope for stateless beans.

Example :

https://github.com/topscode/java/tree/master/module-7/7.2springscope

Introduction of DI

- Dependency Injection (DI) is a design pattern that removes the dependency from the programming code so that it can be easy to manage and test the application.
- Dependency Injection makes our programming code loosely coupled.
- Every java based application has a few objects that work together to present what the end-user sees as a working application.
- When writing a complex Java application, application classes should be as independent as possible of other Java classes to increase the possibility to reuse these classes and to test them independently of other classes while doing unit testing. To understand the DI better,
- Let's understand the Dependency Lookup (DL) first:

Introduction

- The Dependency Lookup is an approach where we get the resource after demand. There can be various ways to get the resource for example:

  T obj = new TImpl();

- In such way, we get the resource (instance of T class) directly by new keyword. Another way is factory method:

  T obj = T.getT();

- This way, we get the resource (instance of T class) by calling the static factory method getT().
- Alternatively, we can get the resource by JNDI (Java Naming Directory Interface) as:

Introduction

```
Context ctx = new InitialContext();
Context environmentCtx = (Context) ctx.lookup("java:comp/env");
A obj = (A)environmentCtx.lookup("T");
```

- The Dependency Injection is a design pattern that removes the dependency of the programs. In such case we provide the information from the external source such as XML file. It makes our code loosely coupled and easier for testing. In such case we write the code as:

Introduction

```
class Employee{
    Address address;
    Employee(Address address){
    this.address=address;
    }
    public void setAddress(Address address){
    this.address=address;
    }
}
```

- In such case, instance of Address class is provided by external souce such as XML file either by constructor or setter method.
- Two ways to perform Dependency Injection in Spring framework:
  - By Constructor
  - By Setter method

DI - Constructor Based

- We can inject the dependency by constructor. The <constructor-arg> sub element of <bean> is used for constructor injection. Here we are going to inject
  - primitive and String-based values
  - Dependent object (contained object)
  - Collection values and objects

Example :
https://github.com/TopsCode/Java/tree/master/Module-7/7.5DiByConstructor

DI – Setter-Getter Based

- We can inject the dependency by setter method also. The <property>subelement of <bean> is used for setter injection. Here we are going to inject
  - primitive and String-based values
  - Dependent object (contained object)
  - Collection values and objects.

Example :
https://github.com/topscode/java/tree/master/module-7/7.4dibyobject

Topics

- Spring Injections
  - Introduction - Inner Beans, Aliases and ID-ref
  - Collections and References
  - Auto Wiring

Introduction

- Inner Beans Alias and IdRef
- Java inner classes are defined within the scope of other classes, similarly, inner beansare beans that are defined within the scope of another bean.
- Thus, a <bean/> element inside the <property/> or <constructor-arg/> elements is called inner bean.

Program using Constructor

Example :

https://github.com/TopsCode/Java/tree/master/Module-7/7.17%20SpringInnerBean1

https://github.com/TopsCode/Java/tree/master/Module-7/7.18%20SpringInnerBean2

https://github.com/TopsCode/Java/tree/master/Module-7/7.19%20SpringInnerBean3

Program using Setter-Getter

Example :

https://github.com/TopsCode/Java/tree/master/Module-7/7.20%20SpringInnerBean4

https://github.com/TopsCode/Java/tree/master/Module-7/7.21%20SpringInnerBean5

Collection and References

- You have seen how to configure primitive data type using value attribute and object references using ref attribute of the <property> tag in your Bean configuration file.
- Both the cases deal with passing singular value to a bean.
- Now what about if you want to pass plural values like Java Collection types List, Set, Map, and Properties.
- To handle the situation, Spring offers four types of collection configuration elements which are as follows:

Collection and References

Element	Description
<list>	This helps in wiring ie injecting a list of values, allowing duplicates.
<set>	This helps in wiring a set of values but without any duplicates.
<map>	This can be used to inject a collection of name-value pairs where name and value can be of any type.
<props>	This can be used to inject a collection of name-value pairs where the name and value are both Strings.

Example :

https://github.com/TopsCode/Java/tree/master/Module-7/7.8%20SpringCollectionList

https://github.com/TopsCode/Java/tree/master/Module-7/7.9%20SpringCollectionBySet/SpringCollectionBySet

https://github.com/TopsCode/Java/tree/master/Module-7/7.10%20SpringCollectionByMap/SpringCollectionByMap

https://github.com/TopsCode/Java/tree/master/Module-7/7.11%20SpringCollectionByProps/SpringCollectionByProps

Auto-Wiring

- Autowiring feature of spring framework enables you to inject the object dependency implicitly.
- It internally uses setter or constructor injection.
- **Autowiring can't be used to inject primitive and string values**.
- It works with reference only.
- The Spring container can autowire relationships between collaborating beans without using <constructor-arg> and <property> elements which helps cut down on the amount of XML configuration you write for a big Spring based application.

Auto-Wiring Modes

Mode	Description
no	It is the default autowiring mode. It means no autowiring bydefault.
byName	The byName mode injects the object dependency according to name of the bean. In such case, property name and bean name must be same. It internally calls setter method.
byType	The byType mode injects the object dependency according to type. So property name and bean name can be different. It internally calls setter method.
constructor	The constructor mode injects the dependency by calling the constructor of the class. It calls the constructor having large number of parameters.
autodetect	Spring first tries to wire using autowire by constructor, if it does not work, Spring tries to autowire by byType. It is deprecated since Spring 3

Program - Constructor

Example :

https://github.com/TopsCode/Java/tree/master/Module-7/7.6AutoWireByConstructor

Program - byName

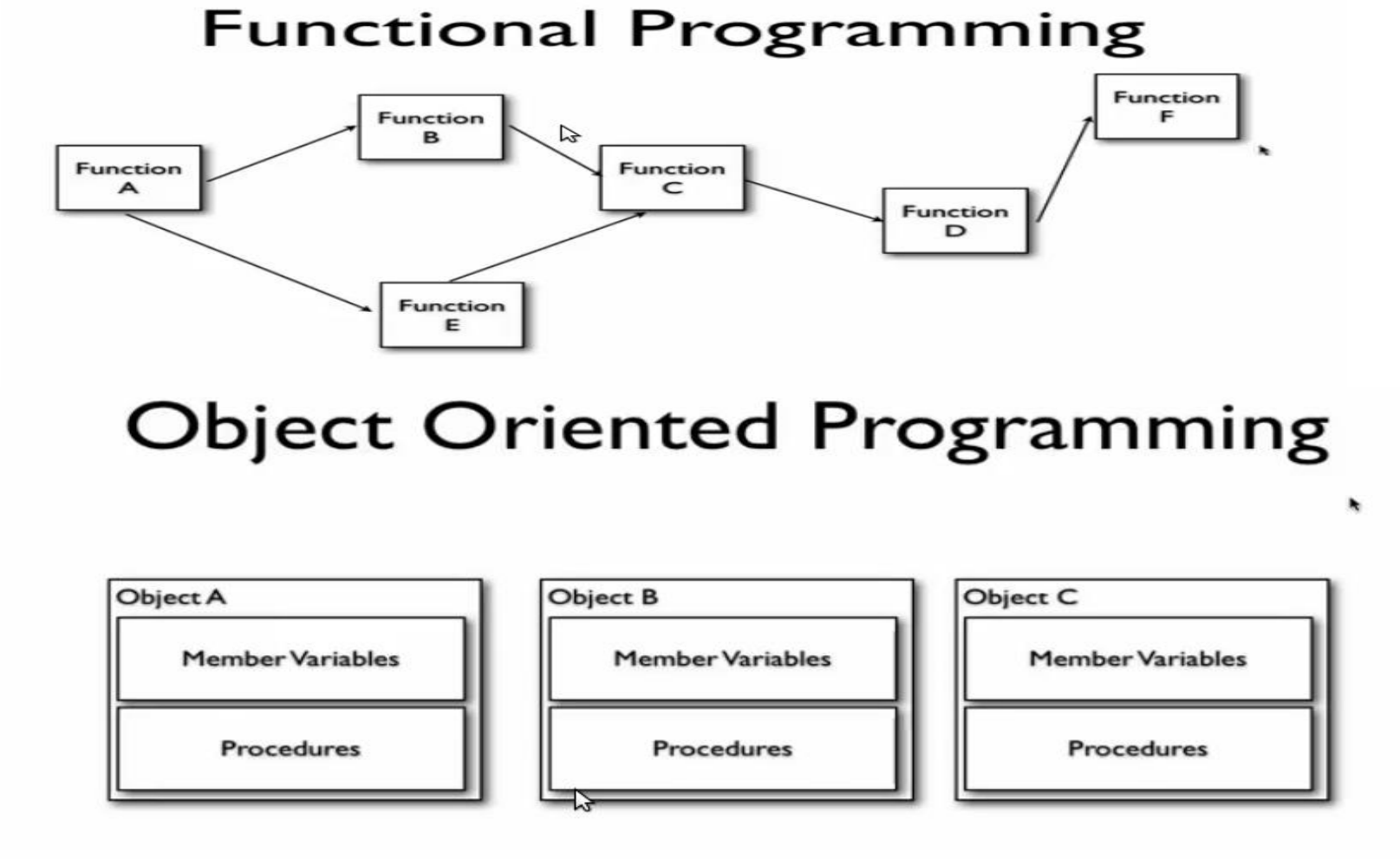
Example :

https://github.com/TopsCode/Java/tree/master/Module-7/7.7%20AutoWireByName

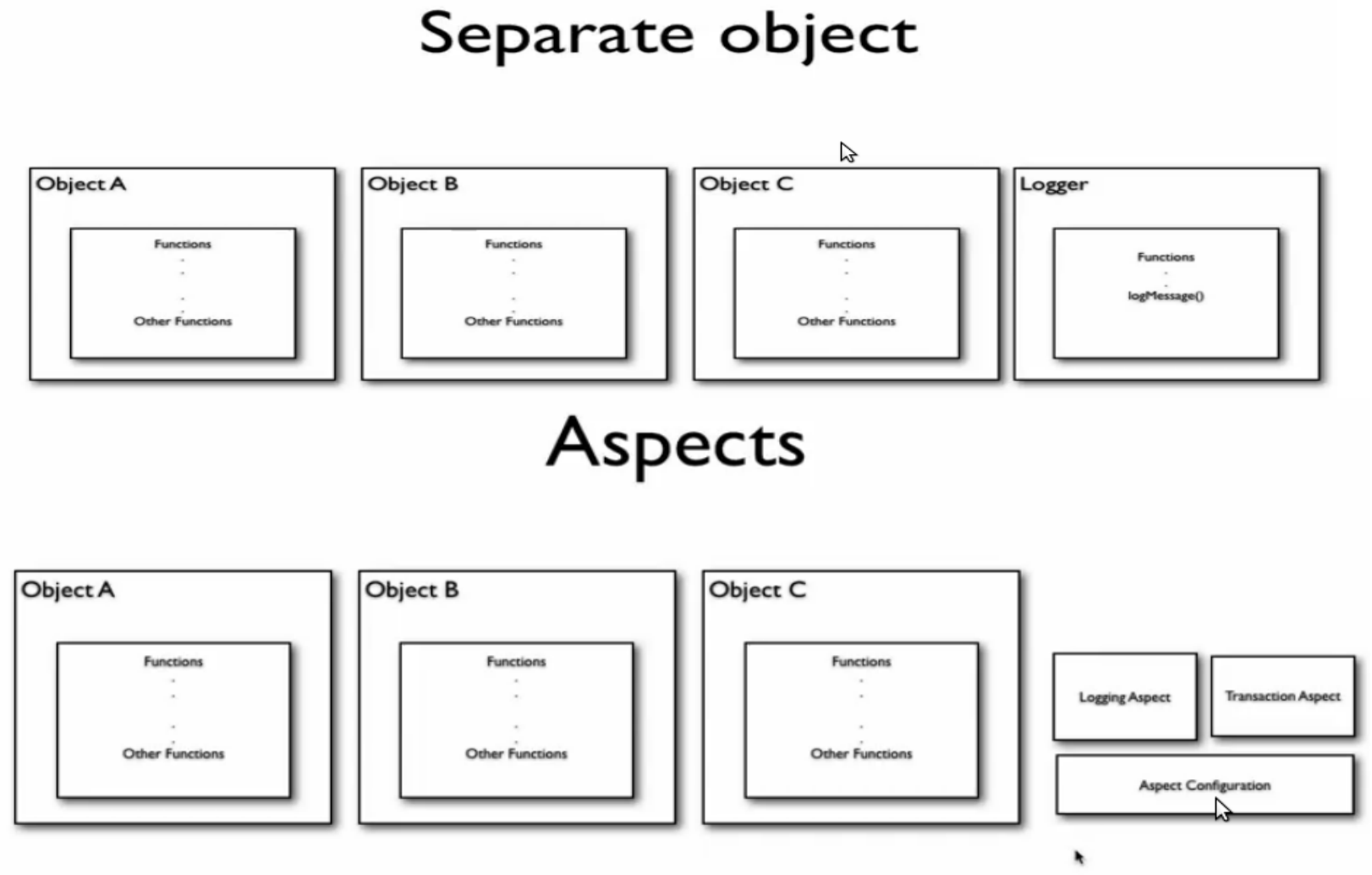
Topics

- Spring Aspect Oriented Programming
  - Introduction
  - AOP Term
  - Write the Aspects

Introduction



Introduction



Spring AOP

- Aspect Oriented Programming (AOP) compliments OOPs in the sense that it also provides modularity.
- But the key unit of modularity is aspect than class.
- Aspect Oriented Programming entails breaking down program logic into distinct parts called so-called concerns.
- AOP breaks the program logic into distinct parts (called concerns).
- It is used to increase modularity by cross-cutting concerns.

Spring AOP

- A cross-cutting concern is a concern that can affect the whole application and should be centralized in one location in code as possible, such as transaction management, authentication, logging, security etc.
- It provides the pluggable way to dynamically add the additional concern before, after or around the actual logic.
- AOP is mostly used in following cases:
  - to provide declarative enterprise services such as declarative transaction management.
  - It allows users to implement custom aspects.

Example :
https://github.com/TopsCode/Java/tree/master/Module-7/7.22%20Spring_AOP

Spring AOP

Terms	Description
Aspect	It is a class that contains advices, joinpoints etc.
Join point	Join point is any point in your program such as method execution, exception handling, field access etc. Spring supports only method execution join point.
Advice	Advice represents an action taken by an aspect at a particular join point. There are different types of advices: • **Before Advice:** it executes before a join point. • **After Returning Advice:** it executes after a joint point completes normally. • **After Throwing Advice:** it executes if method exits by throwing an exception. • **After (finally) Advice:** it executes after a join point regardless of join point exit whether normally or exceptional return. • **Around Advice:** It executes before and after a join point.
Pointcut	It is an expression language of AOP that matches join points.
Introduction	It means introduction of additonal method and fields for a type. It allows you to introduce new interface to any advised object.
Target object	It is the object i.e. being advised by one or more aspects. It is also known as proxied object in spring because Spring AOP is implemented using runtime proxies.

Spring AOP

Terms	Description
Interceptor	It is an aspect that contains only one advice.
AOP Proxy	It is used to implement aspect contracts, created by AOP framwork. It will be a JDK dynamic proxy or CGLIB proxy in spring framework.
Weaving	It is the process of linking aspect with other application types or objects to create an advised object. Weaving can be done at compile time, load time or runtime. Spring AOP performs weaving at runtime.

Spring ORM

- Spring provides API to easily integrate Spring with ORM frameworks such as Hibernate, JPA(Java Persistence API), JDO(Java Data Objects), Oracle Toplink and iBATIS.
- We can simply integrate hibernate application with spring application.
- In hibernate frameworks, we provide all the database information hibernate.cfg.xml file.
- But if we are going to integrate the hibernate application with spring, we don't need to create the hibernate.cfg.xml file.
- We can provide all the information in the Bean.xml (applicationContext.xml) file.

Advantages

- The Spring framework provides HibernateTemplate class, so you don't need to follow so many steps like create Configuration, BuildSessionFactory, Session, beginning and committing transaction etc. So it saves a lot of code.

Advantages	
Less coding is required	By the help of Spring framework, you don't need to write extra codes before and after the actual database logic such as getting the connection, starting transaction, committing transaction, closing connection etc.
Easy to test:	Spring's IoC approach makes it easy to test the application
Better exception handling	Spring framework provides its own API for exception handling with ORM framework.

Advantages

Advantages	
Integrated transaction management	By the help of Spring framework, we can wrap our mapping code with an explicit template wrapper class or AOP style method interceptor.

Example :
https://github.com/topscode/java/tree/master/module-7/7.13%20springorm

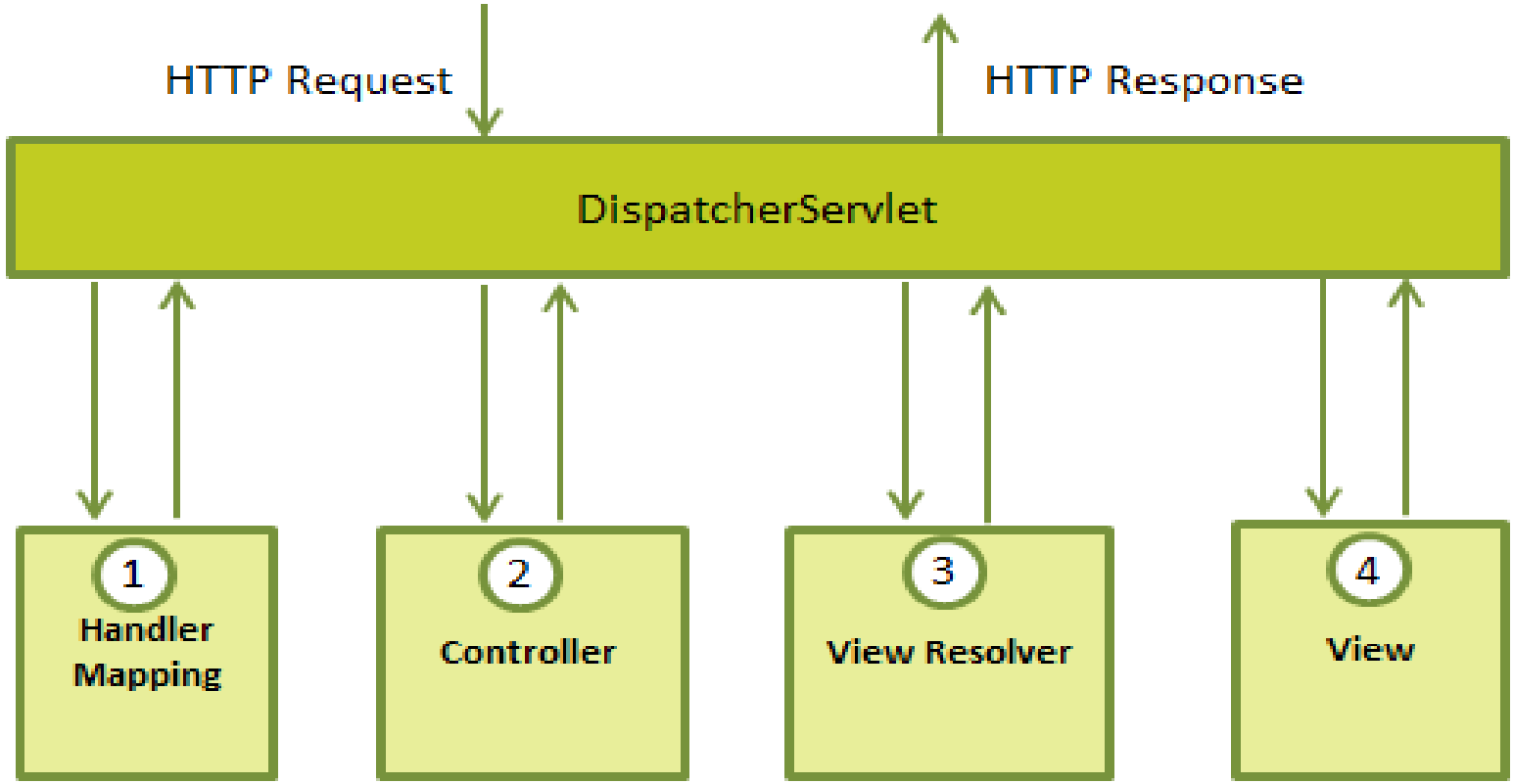
Spring MVC Framework

- The Spring web MVC framework provides model-view-controller architecture and ready components that can be used to develop flexible and loosely coupled web applications.
- The MVC pattern results in separating the different aspects of the application (input logic, business logic, and UI logic), while providing a loose coupling between these elements.
  - The **Model** encapsulates the application data and in general they will consist of POJO.
  - The **View** is responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.
  - The **Controller** is responsible for processing user requests and building appropriate model and passes it to the view for rendering.

Request Flow of Spring MVC

- The Spring Web model-view-controller (MVC) framework is designed around a DispatcherServlet that handles all the HTTP requests and responses. The request processing workflow of the Spring Web MVC DispatcherServlet is illustrated in the following diagram:

Request Flow of Spring MVC



Request Flow of Spring MVC

- After receiving an HTTP request, DispatcherServlet advice the HandlerMapping to call the appropriate Controller.
- The Controller takes the request and calls the appropriate service methods based on used GET or POST method.
  - The service method will set model data based on defined business logic and returns view name to the DispatcherServlet.
  - The service method will set ModelAndView object to define the model data, view name and commander name of the form data.
- The DispatcherServlet will take help from ViewResolver to pickup the defined view for the request.
- Once view is finalized, The DispatcherServlet passes the model data to the view which is finally rendered on the web browser.

Example :
https://github.com/TopsCode/Java/tree/master/Module-7/7.12%20SpringMVC

Spring Page Redirections

- The following example show how to write a simple web based application which makes use of redirect to transfer a http request to another page.
- To start with it, let us have working Eclipse IDE in place and follow the following steps to develop a Dynamic Form based Web Application using Spring Web Framework:

Example :
https://github.com/TopsCode/Java/tree/master/Module-7/7.12%20SpringMVC

Spring Static Page Redirection

- The following example show how to write a simple web based application using Spring MVC Framework, which can access static pages along with dynamic pages with the help of <mvc:resources> tag.
- To start with it, let us have working Eclipse IDE in place and follow the following steps to develope a Dynamic Form based Web Application using Spring Web Framework:

Example :

https://github.com/TopsCode/Java/tree/master/Module-7/7.12%20SpringMVC

Spring MVC Web Forms

- **Note:**
- In BeanController we are define 2 different method for two different way to passing the parameters using forms. You can use any of way.
  - First way is ModelAndView technique to return the object of ModelAndView by the passing view name, commander name, and its object in constructor.
  - Second way to pass multiple primitive and object values by using the Model interface object by setting the attributes and return the view name as string parameter.

Spring Form Tags - Validation

- The following example show how to write a simple web based application which makes use of Form tag library with dynamic data populating and static data using Spring Web MVC framework.
- To start with it, let us have working Eclipse IDE in place and follow the following steps to develop a Dynamic Form based Web Application using Spring Web Framework:

Example :

https://github.com/TopsCode/Java/blob/master/Module-7/7.16%20SpringForm/WebContent/index.jsp

Session Management

- Session management is one of the essential parts for each web application.
- Since Spring MVC is a powerful framework for web development, it has its own tools and API for the interaction with sessions.
- This is a simple Spring MVC controller with the one extra @SessionAttributes annotation.
- It indicates that in the controller's methods some values can be assigned to the arguments of the annotation.
- In this example I have declared just one session attribute with the name "login".
- That means I can put some object into ModelAndView using the addObject() method, and it will be added to the session if the name of the object will be the same as the name of the argument in @SessionAttributes. Example :

https://github.com/topscode/java/tree/master/module-7/7.14%20sessionmanagement

CRUD Operation with JDBC

- In Spring MVC, we are performing CRUD (Create, Read , Update, Delete) operation by using JDBCTemplate and HibernateTemaplte. So We can performing to integration of JDBC API and Hibernate.
- **Spring CRUD Operation Steps**

1. To add DispatcherServlet servlet configuration in web.xml file
2. To add spring-servlet.xml file with Bean configuration of InternalResourceViewResolver. component scanning and mvc annotation driver.
3. Create Bean POJO Class with properties and their getter and setter.
4. If you are using hibernate them create bean.hbm.xml file.
5. Create Controller with index method.
6. Create index.jsp page with Adding the entery form.
7. Create Validation of the Bean with Validator Class
8. Create validator properties file
9. To add validator bean mapping and ResourceBindleMessageSource for properties configuration in spring-servlet.xml file.

CRUD Operation with JDBC

10. Create DAO and Service interface with declaring the methods.
11. Create DAOImpl and SerivceImpl clases for implementing the interfaces.
12. To add DAO and Service bean mapping in spring-servlet.xml file.
13. To add datasource mapping in spring-servlet.xml file.
14. If you are using hibernate then you have to configure session factory using LocalSessionFactoryBean with hibernate properties and resource mapping files.
15. To add method in controller for inserting data in database via service calling after validating form bean data.
16. Redirect to control to show method for setting attribute list data values of bean. It will show the data entries in show.jsp page.
17. To click on delete button to call delete method of controller for deleting the entries via service method calling. Afterword redirect control to show.jsp via show method of controller for setting the list attribute.
18. To click on edit link to call edit method of controller for fetching specific bean object via service method and set ModelAndView with commander object. Show entries in edit.jsp page for updating entries.
19. To update entries in database by update method of controller. This update method call though edit.jsp page form. After redirect to show.jsp via call show method through update method.

Example :

https://github.com/TopsCode/Java/tree/master/Module-7/7.15%20SpringMVCORM

Spring Boot

- Spring Boot is a Spring module which provides RAD (Rapid Application Develop

feature to Spring framework.

- It is used to create stand alone spring based application that you can just run because it needs very little spring configuration.

- Spring Boot does not generate code and there is absolutely no requirement for XML configuration.

- It uses convention over configuration software design paradigm that means it decrease the effort of developer.

Advantages

- Create stand-alone Spring applications that can be started using java -jar.
- Embed Tomcat, Jetty or Undertow directly. You don't need to deploy WAR files.
- It provides opinionated 'starter' POMs to simplify your Maven configuration.
- It automatically configure Spring whenever possible.
- It provides production-ready features such as metrics, health checks and externalized configuration.
- Absolutely no code generation and no requirement for XML configuration.

Spring Boot Project

- There are multiple approaches to create Spring Boot project. We can use any of the following approach to create application.

- Spring Maven Project

- Spring Starter Project Wizard

- Spring Initializr

- Spring Boot CLI

Example :
https://github.com/topscode/java/tree/master/module-7/7.23%20springboot

Module 8
Application Deployment on Cloud

Module 8
Application Deployment on Cloud

- Introduction
- Types
- Hosting
  - (How to Deploy JAVA web application With Cloud)

TOPS Technologies

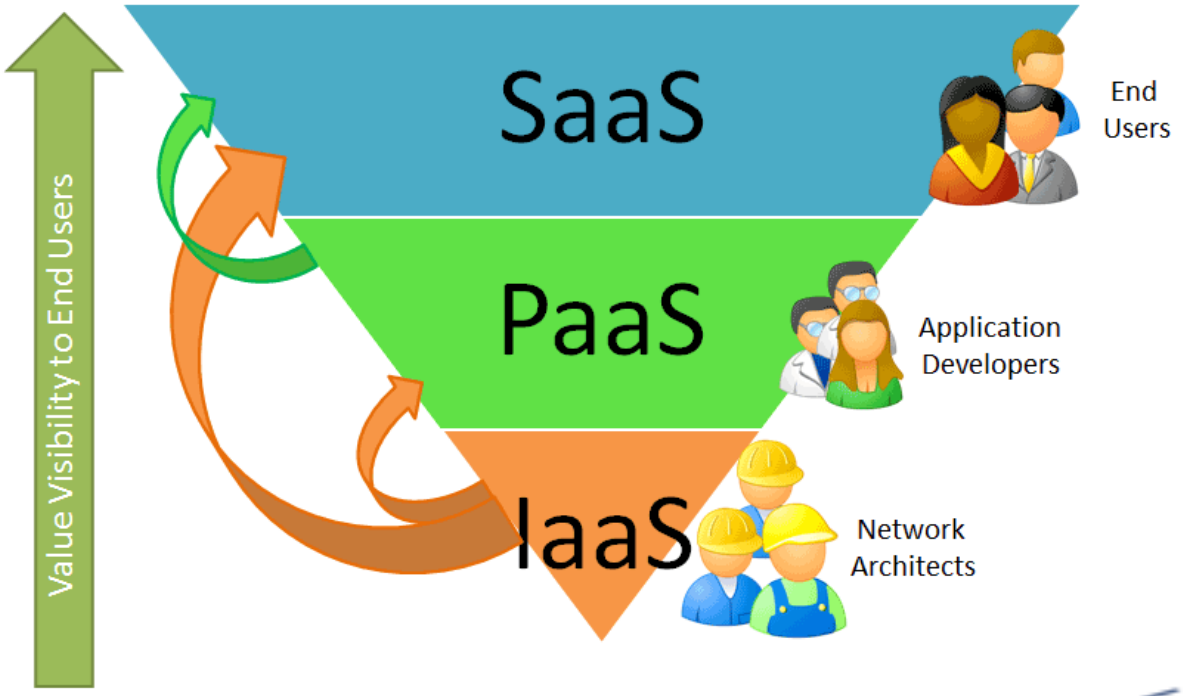
What is Cloud Computing?

- The “cloud” in cloud computing can be defined as the set of hardware, networks, storage, services, and interfaces that combine to deliver aspects of computing as a service.

- Cloud services include the delivery of software, infrastructure, and storage over the Internet (either as separate components or a complete platform) based on user demand.

TOPS Technologies

Cloud Computing Models



TOPS Technologies

Infrastructure as a Service (IaaS)

- The IaaS layer offers storage and compute resources that developers and IT organizations can use to deliver business solutions.

Platform as a Service (PaaS)

- The PaaS layer offers black-box services with which developers can build applications on top of the compute infrastructure.
- This might include developer tools that are offered as a service to build services, or data access and database services, or billing services.

Software as a Service (SaaS)

- In the SaaS layer, the service provider hosts the software so you don’t need to install it, manage it, or buy hardware for it.
- All you have to do is connect and use it. SaaS Examples include customer relationship management as a service.

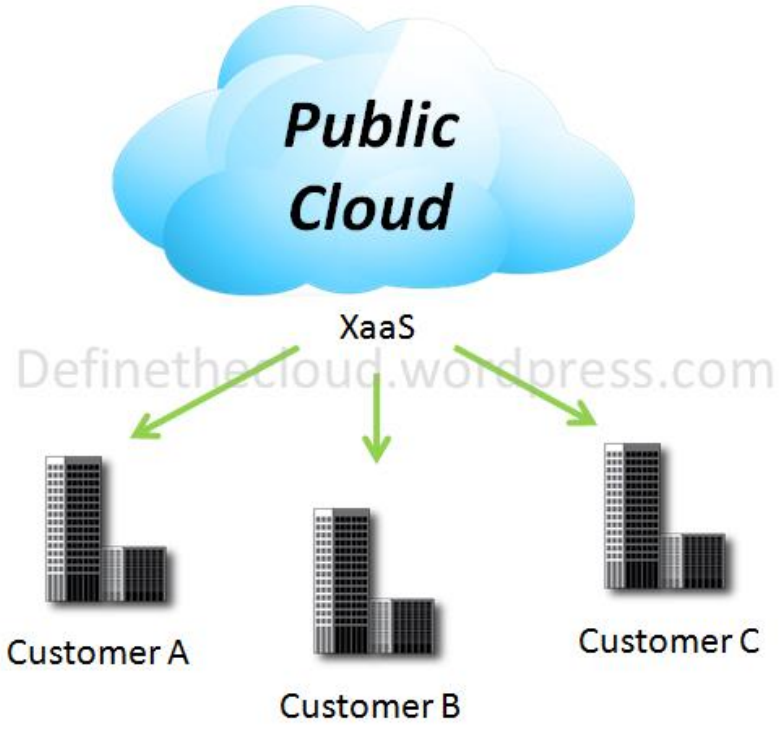
Deployment of Cloud Service

- Public Cloud
- Private Cloud
- Community Cloud
- Hybrid Cloud

TOPS Technologies

Public Cloud

- A public cloud is one in which the services and infrastructure are provided off-site over the Internet.
- These clouds offer the greatest level of efficiency in shared resources; however, they are also more vulnerable than private clouds.

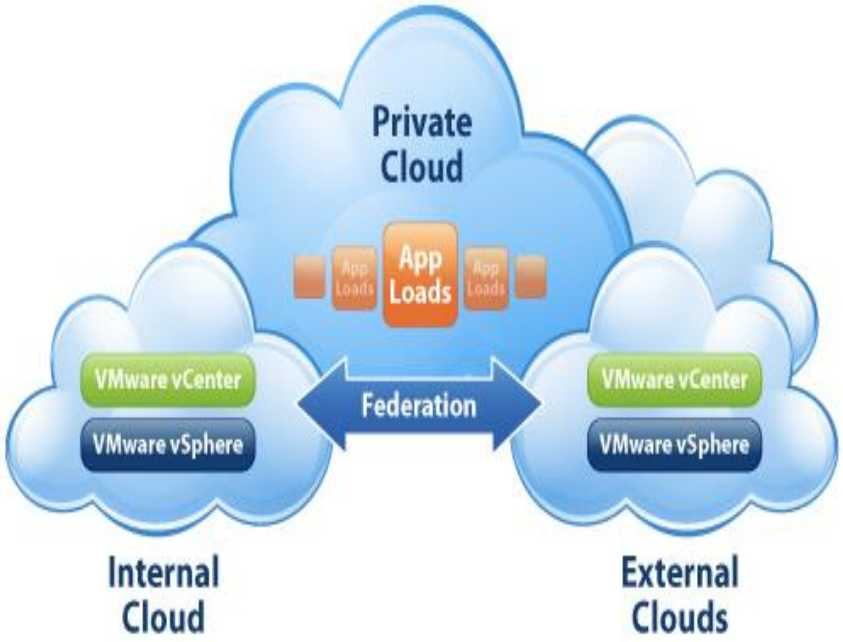


A public cloud is the obvious choice when

- Your standardized workload for applications is used by lots of people, such as e-mail.
- You need to test and develop application code.
- You have SaaS (Software as a Service) applications from a vendor who has a well-implemented security strategy.
- You need incremental capacity (the ability to add computer capacity for peak times).
- You’re doing collaboration projects.
- You’re doing an ad-hoc software development project using a Platform as a Service (PaaS) offering cloud.

Private Cloud

- A private cloud is one in which the services and infrastructure are maintained on a private network.
- These clouds offer the greatest level of security and control, but they require the company to still purchase and maintain all the software and infrastructure,
- which reduces the cost savings



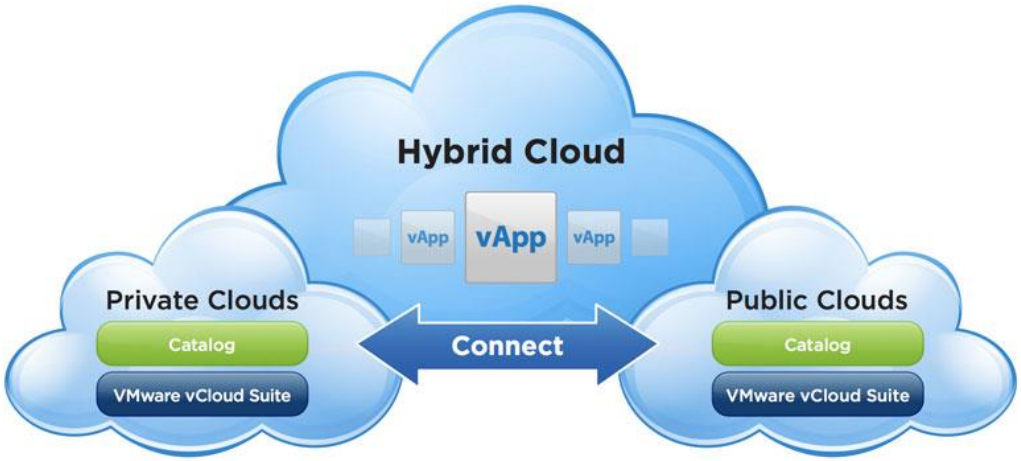
A private cloud is the obvious choice when

- Your business is your data and your applications. Therefore, control and security are paramount.
- Your business is part of an industry that must conform to strict security and data privacy issues.
- Your company is large enough to run a next generation cloud data center efficiently and effectively on its own.

TOPS Technologies

Hybrid Clouds

- A hybrid cloud includes a variety of public and private options wit multiple providers. By spreading things out over a hybrid cloud, y keep each aspect at your busines in the most efficient environmen possible.
- The downside is that you have t keep track of multiple different security platforms and ensure tha all aspects of your business can communicate with each other.



TOPS Technologies

A Hybrid cloud is the obvious choice when

- Your company wants to use a SaaS application but is concerned about security. Your SaaS vendor can create a private cloud just for your company inside their firewall. They provide you with a virtual private network (VPN) for additional security.
- Your company offers services that are tailored for different vertical markets. You can use a public cloud to interact with the clients but keep their data secured within a private cloud.

TOPS Technologies

How To Deployed JAVA Application on Cloud

- Create an "Application" in OpenShift (With the command-line or via their IDE)
- Code the application (in Vi, TextMate, Eclipse, Visual Studio, or whatever)
- Push the application code to OpenShift(again, with the command-line or from their IDE)

OPENSHIFT

Steps for Application Deployment

1. Sign Up Open shift account and verify the account with your e-mail

2. Login The Account with Your Username and Password

3. To create an application from the Create Application-Menu Item

TOPS Technologies

Thank You

TOPS Technologies