

CORE JAVA

Programming language

A language which is used to communicate between user and computer is called programming language.

Programming language acts like a mediator or interface between user and computer.

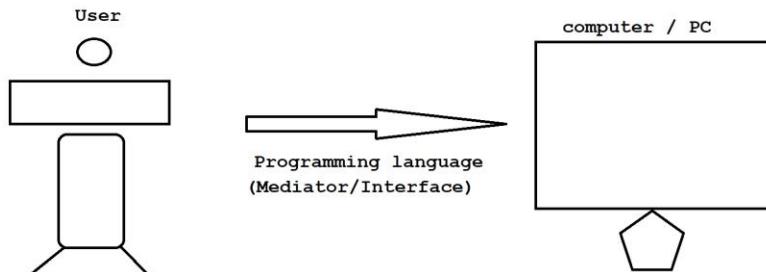


Diagram: introduction1.1

Java

- Object oriented programming language.
- Platform independent programming language.
- Case sensitive programming language
- Strongly typed checking language.
- High level programming language.
- Open source programming language

1995 --> James Gosling --> Sun Micro system --> Oracle Corporation Java software --> JDK software

C language

- Procedure oriented programming language.
- Platform dependent programming language.
- Case sensitive programming language
- Loosely typed checking language.
- Middle level language (LOW + HIGH)

Interview Questions

Q) What is Java?

Java is an object oriented, platform independent, case sensitive, strongly typed checking, high level, open source programming language developed by James Gosling in the year of 1995.

Programming language

A language which is used to communicate between user and computer is called programming language.

Programming language acts like a mediator or interface between user and computer.

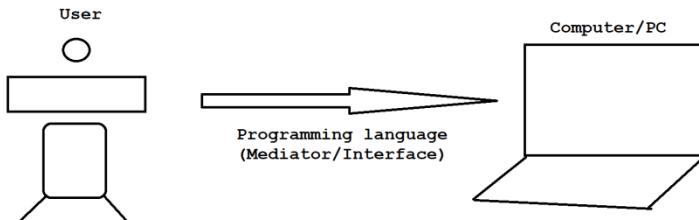


Diagram: introduction2.1

Programming language is divided into two types.

1. Low Level Language
2. High Level Language

1. Low Level Language

A language which is understand by a computer easily is called low level language.

A language which is computer dependent is called low level language.

ex: Machine language
 Assembly language

Machine language

It is a fundamental language of a computer which is combination of 0's and 1's.

It is also known as binary language.

A computer understands many languages but to understand machine language computer does not required any translator.

Advantages:

- > A program written in machine language consumes less memory.
- > It does not require any translator.
- > It is more efficient when compared to other languages.

Disadvantages:

- > It is a burden on a programmer to remember dozens of binary code.
- > If anywhere error raised in our program then locating and handling that becomes difficult.
- > Modifications we can't do easily.

Assembly language

The second generation language came into existence is called assembly language.

Assembly language is a replacement of symbols and letters for mathematical programming code i.e. opcode values.

It is also known as symbolic language.

Assembly language can't understand by a computer directly. We required translator.

We have three translators.

- 1) Assembler
- 2) Compiler
- 3) Interpreter

1) Assembler

It is one of the translator which is used to convert assembly code to machine code.

Merits:

- > If anywhere error raised in our program then locating and handling that error becomes easy.
- > Modifications can be done easily.

Demerits:

- > It is a mind trick to remember all symbolic code.
- > It requires translator.
- > It is less efficient when compare to machine language.

Q) What is Debugging?

Bug is also known as Error.

The process of eliminating the bugs from the application is called debugging.

2. High Level Language

A language which is understand by a user easily is called high level language.

A language which is user dependent is called high level language.

ex: C++, C#, Java, .Net, Python and etc.

High level language can't understand by a computer easily. We required translator.

compiler It will compile and execute our program at a time.

Interpreter It will execute our program line by line procedure.

Advantages:

- > It is easy to learn and easy to use because it is similar to english language.
- > Debugging can be done easily.
- > Modifications can be done easily.

Disadvantages:

- > A program written in high level language comes huge amount of memory.
- > It requires translator.
- > It is not efficient when compare to low level language.

Escape Characters or Escape Sequences

Escape characters are used to design our output in neat and clean manner.

All escape characters starts with back slash() followed by a single character.

ex: \n

Mostly escape characters are placed inside output statement in java.

ex: System.out.println("\n");

We have following escape characters or sequences in java.

- 1)\n (new line)
- 2)\t (horizontal tab)
- 3)\b (back space)
- 4)\r (carriage return)
- 5)\f (form feeding)
- 6)\\ (back slash)

7)\" (double quote)

8)\' (single quote) and etc.

1)\n (new line)

class Dhanush

{

```
    public static void main(String[] args)
    {
        System.out.println("IHUB\nTALENT");
    }
```

}

o/p:

```
    IHUB
    TALENT
```

2)\t (horizontal tab)

class Chiranjeevi

{

```
    public static void main(String[] args)
    {
        System.out.println("IHUB\tTALENT");
    }
```

}

o/p:

```
    IHUB  TALENT
```

3)\b (back space)

class Divya

{

```
    public static void main(String[] args)
    {
        System.out.println("IHUBTAL\bENT");
    }
```

}

o/p:

```
    IHUBTAENT
```

ex:

class Kalyani

{

```
    public static void main(String[] args)
    {
        System.out.println("IHUB\b\b\bTALENT");
    }
```

}

o/p: ITALENT

4)\r (carriage return)

```
class Gopala
{
    public static void main(String[] args)
    {
        System.out.println("IHUB\rTALENT");
    }
}
```

o/p:
TALENT

ex:

```
class Karthik
{
    public static void main(String[] args)
    {
        System.out.println("TALENT\rIHUB");
    }
}
```

o/p:
IHUBNT

6)\\ (back slash)

```
class Saritha
{
    public static void main(String[] args)
    {
        System.out.println("IHUB\\TALENT");
    }
}
```

o/p:
IHUB\TALENT

7)\" (double quote)

```
class Sufiyan
{
    public static void main(String[] args)
    {
        System.out.println("IHUB\"TALENT");
    }
}
```

o/p:
IHUB"TALENT

8)\' (single quote)

```
class Vyshnavi
{
    public static void main(String[] args)
    {
        System.out.println("IHUB'TALENT");
        System.out.println("IHUB\'TALENT");
    }
}
```

o/p: IHUB'TALENT
IHUB\'TALENT

Interview question

Q)Write a c program to print %d ?

```
void main()
{
    clrscr();

    printf("%d"); // 0
```

```
    getch();
}
```

ex:

```
void main()
{
    clrscr();

    printf("%%d"); // %d

    getch();
}
```

Q)What is the output of below program?

```
class Demo
{
    public static void main(String[] args)
    {
        System.out.print("\nek");
        System.out.print("\boi");
        System.out.print("\rha");
    }
}
```

o/p:
hai

SDLC

SDLC stands for Software Development Life Cycle.

It is a process adopted by IT industry to develop accurate and quality of softwares.

There are six phases in SDLC.

- 1) Feasibility study
- 2) Analysis
- 3) Designing
- 4) Coding
- 5) Testing
- 6) Delivery and Maintenance

1) Feasibility study

Feasibility study completed depends upon TELOS formulae.

ex: T - Technical Feasibility
E - Economic Feasibility
L - Legal Feasibility
O - Operational Feasibility
S - Scheduled Feasibility

All the above information they will keep in a document called BDD.

BDD stands for Business Designed Document.

2) Analysis

In this phase, system analyst or product owner will be involved.

They will separate system requirements and software requirements.

They will keep this information in a document called SRS.

SRS stands for Software Requirement Specification.

3) Designing

Designing can be performed in two ways.

i) High level designing

Manager is responsible to perform high level designing.

In high level designing, we will design main modules.

ii) Low level designing

Team Lead/Project Lead is responsible to perform low level designing.

In low level designing, we will design sub-module/child modules.

All this above information we will keep in a document called PDD/TDD.

Here PDD stands for Project Design Document.

Here TDD stands for Technical Design Document.

4) Coding

In coding phase, developers will be involved.

Developers are responsible to generate the build(source code).

Once our build is ready then developer even responsible to perform white box testing.

5) Testing Phase

In this phase, testing team or QA team will be involved.

The will test the code by using one software component called STLC.

STLC stands for Software Testing Life Cycle.

Testing Team or QA team is responsible to perform black box testing.

6) Delivery & maintenance phase

Before delivery we will perform UAT testing.

UAT stands for User Acceptance Testing.

UAT testing is classified into two types.

i) Alpha testing

ii) Beta testing

Project

Project is a collection of modules.

ex:

customer module

login module

registration module

payment module

report generation module

and etc.

Every project contains two domains.

1) Technical Domain

Using which technology we developed our project.

ex: Java

2) Functional Domain

It describes state of a project.

ex: Healthcare domain

Insurance domain

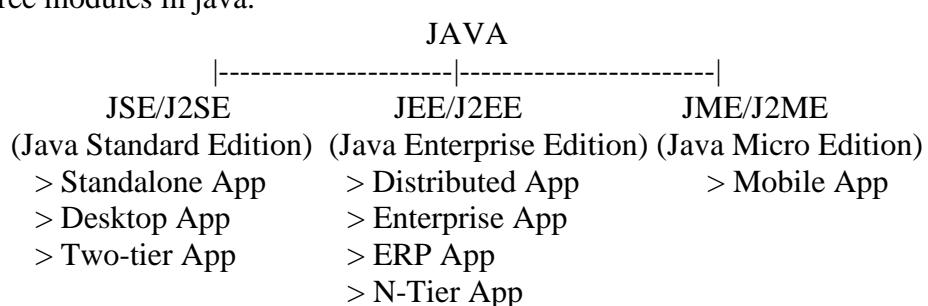
Banking domain

ERP domain

and etc.

Modules In Java

There are three modules in java.



> Standalone App

A normal java program which contains main method is called standalone application.
ex:

```
class Test
{
    public static void main(String[] args)
    {
        -
        -
    }
}
```

> Desktop App

It is a software application which is used to perform particular task.

It is specially designed for laptops and PC's.

ex: Control panel, Recycle Bin

> Two-tier App

Having more than one tier is called two-tier application.

Diagram: java1.1

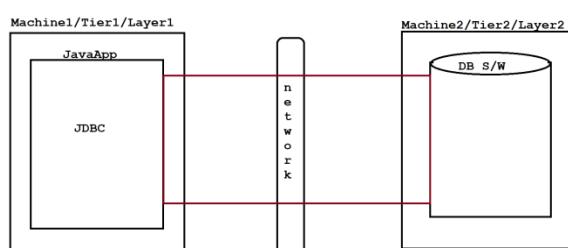
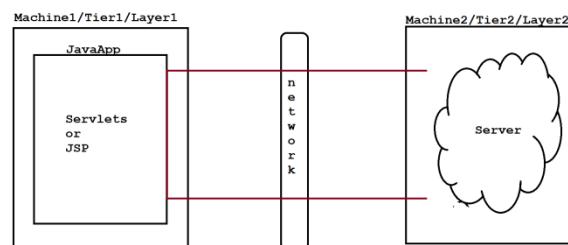


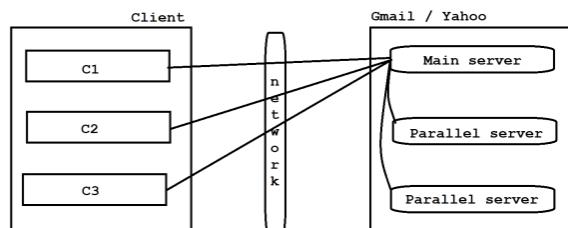
Diagram: java1.2



> Distributed App

In client-server architecture, if multiple clients are giving the request to the main server then main server will forward the request to its parallel servers to reduce the burden of main server such type of application is called distributed application.

Diagram: java1.3



> Enterprise App

An application which deals with large business complex logic with the help of middleware services is called enterprises application.

Here middleware services means authentication, authorization, malware production, firewall, security and etc.

ex: facebook

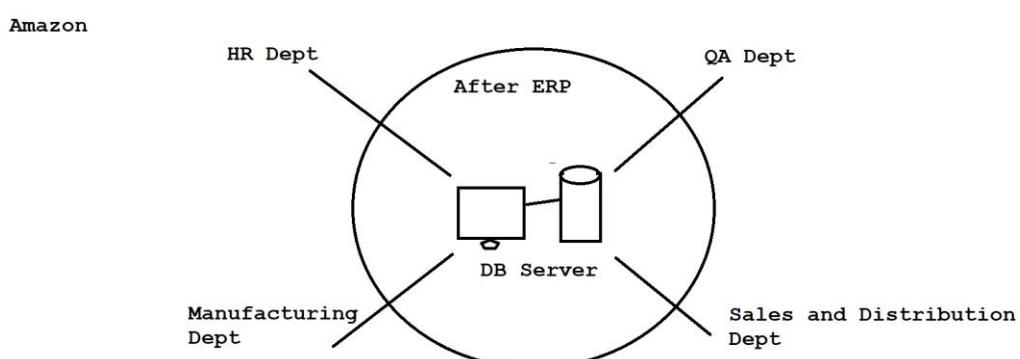
Online shopping websites

> ERP App

ERP stands for Enterprise Resource Planning.

ERP application is used to maintain the data in a enterprise.

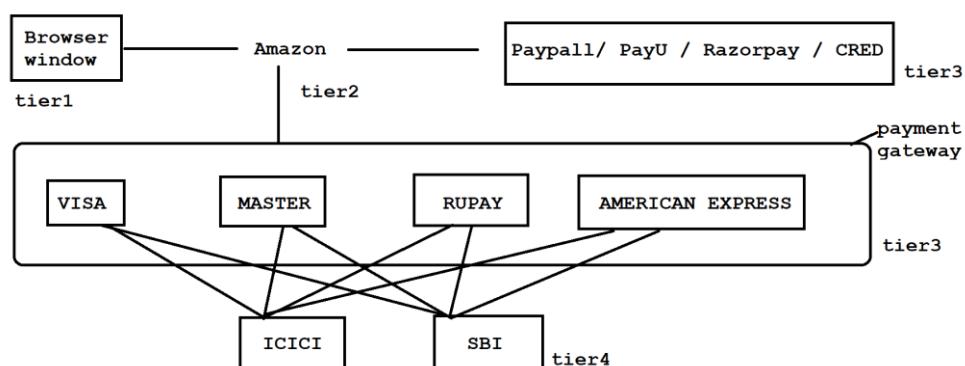
Diagram: java1.4



> N-Tier App

Having more then two tiers is called N-tier application.

Diagram: java1.5



> Mobile App

It is a software application or a program which is specially designed for wireless network devices like phone, tab, cell, cellular and etc.

ex:

Gpay
PhonePay
TempleRun
and etc.

Comments in java

Comments are created for documentation purpose.

Comments are used to improve readability of our code.

It is highly recommended to use comments in our regular programming.

Comments will not display in output because our compiler does not compile the comments.

In java we have two types of comments.

1)Single Line Comment

It is used to comment a single line.

ex: // comment here

2)Multiple Line Comment

It is used to comment multiple lines.

ex: /*

-
- comment here
-
*/

ex:

```
//class declaration
class Test
{
    //main method
    public static void main(String[] args)
    {
        //variable declaration
        int i=10;
        //output statement
        System.out.println(i);
    }
}
```

Q)Differences between python and java?

Python

It is developed by Guido Van Rossum.

It is a product of Microsoft.

It is a scripting language.

It is a interpreted language.

It contains PVM.

It is a dynamically typed language.

Performance is low.

There is low security.

It contains less code.

Java

It is developed by James Gosling.

It is a product of Oracle Corporation.

It is object oriented programming language.

It is a compiled language.

It contains JVM.

It is a statically typed language.

Performance is high.

There is high security.

It contains more code.

IT companies

We have two types of IT companies.

1)Service Based company

ex: TCS, Capgemini, Cognizant and etc.

2) Product Based company

ex: Microsoft, Oracle Corporation, IBM, Amazon and etc.

Naming Conventions in java

In java uppercase letters will consider as different and lowercase letters will consider as different. Hence we consider java is a case sensitive programming language.

As java is a case sensitive , we must and should follow naming conventions for following things.

ex: classes
interfaces
variables
methods
keywords
packages
constants

classes

In java, a class name must and should starts with uppercase letter and if it contains multiple words then each inner word starts with initcap.

<u>predefined classes</u>	<u>userdefined classes</u>
System	Test
StringBuilder	DemoApp
FileWriter	ExampleApplication
BufferedReader	QualityThought
and etc.	and etc.

interfaces

In java, an interface name must and should starts with uppercase letter and if it contains multiple words then each inner word starts with initcap.

<u>predefined interfaces</u>	<u>userdefined interfaces</u>
Serializable	ITest
Cloneable	IDemoApp
Enumeration	IExampleApplication
ListIterator	IQualityThought
and etc.	and etc.

variables

In java , a variable name must and should starts with lower case letter and if it contains multiple words then each inner word starts with initcap.

<u>predefined variables</u>	<u>userdefined variables</u>
out	empId
in	studName
err	deptNo
length	serialNo
and etc.	and etc.

methods

In java, a method name starts with lower case letter and if it contains multiple words then each inner word starts with initcap.

ex: predefined methods userdefined methods
 getClass() calculateBill()
 setPriority() getEmployeeDetails()
 setName() setInfo()
 hashCode() getStudentsDetails()
 toString() and etc.
 and etc.

keywords

In java, all keywords we need to write under lowercase letters only.

ex: predefined keywords
 if
 else
 public
 static
 for
 void
 and etc.

packages

In java, all packages we need to write under lower case letters only.

ex: predefined packages userdefined packages
 java.lang com.ihubtalent.www
 java.io com.google.www
 java.util com.facebook.www
 java.text and etc.
 java.time
 java.sql
 and etc.

constants

In java, all constants we need to write under uppercase letters only.

ex: predefined constants userdefined constants
 MAX_PRIORITY LIMIT
 MIN_PRIORITY DATA
 NORM_PRIORITY and etc.
 MAX_VALUE
 MIN_VALUE
 and etc.

Assignment

class : RajKumar
interface : IRajKumar

variable	:	rajKumar
method	:	rajKumar()
package	:	com.rajkumar.www
constant	:	RAJKUMAR or RAJ_KUMAR

Interview Questions

Q) Which is a default package in java?

java.lang package

Q) What is package?

Package is a collection of classes and interfaces.

Q) How many classes are there in java?

Java 7	-	4024 classes
Java 8	-	4240 classes
Java 9	-	6005 classes
Java 10	-	6002 classes

Q) What is Java?

Java is a object oriented, platform independent ,case sensitive, strongly typed checking , high level , open source programming language developed by James Gosling in the year of 1995.

Q) Differences between C++ and Java?

C++

It is developed by Bjarne Stroustrup.

It is a partial object oriented programmin language.

It is a platform dependent.

Memory allocation and deallocation will taken create by a programmer.

It supports multiple inheritance.

It supports pointers.

It supports operator overloading.

It supports preprocessor direcotry(#).

It supports three access specifiers i.e public,private and protected.

It contains three loops i.e do while loop, while loop and for loop.

Java

It is developed by James Gosling.

It is purely object oriented programming language.

It is a platform independent.

Memory allocation and deallocation will taken care by a JVM.

It does not support multiple inheritance.

It does not support pointers.

It does not support operator overloading.

It does not support preprocessor direcotry(#).

It supports four access modifiers i.e default,public,private and protected.

It contains four loops i.e do while loop, while loop, for loop and for each loop.

Q)What are the features of Java?

We have following important features in java.

1. Simple
2. Object oriented
3. Platform independent
4. Highly secured
5. Robust
6. Portable
7. Architecture Neutral
8. Multithreaded
9. Dynamic and etc.

Q)Who is the responsible to destroy objects in java?

Garbage Collector

Q)What is garbage collector ?

Garbage collector is used to destroy unused or useless objects from java.

Q)In how many ways we can call garage collector?

There are two ways to call garbage collector in java.

- 1) System.gc()
- 2) Runtime.getRuntime().gc()

History of Java

In 1990, Sun Micro system took one project to develop a software called consumer electronic device which can be controlled by a remote like setup box. That time project was called Stealth project and later is renamed to Green project.

James Gosling , Mike Sheridan, Patrick Naughton were there to develop this project and they met in a place called Aspan/Colarado to start the work with Graphic System.James Gosling decided to use C and C++ languages to develop this project.But the problem what they have faced is they are system dependent.Then James Gosling decided why don't we create our own programming language which is system independent.

In 1991, they have developed on programming language called an OAK.OAK means strength, itself is a coffee seed name and it is a national tree for many contries like Germany , France , USA and etc. Later in 1995 they have renamed OAK to Java.Java is a island of an Indonesia where first coffee of seed was produced and during the development of project they were consuming lot of coffee's.Hence symbol of java is a cup of coffee with saucer.

Interview questions

Q) Who is the creator of java?

James Gosling

Q) In which year java was developed?

In 1995

Q) Java originally known as ____ ?

OAK

Q) Java is a platform dependent or platform independent?

It is platform independent

Q) Java is a product of which company?

Oracle Corporation

Q) Which is the latest version of Java?

Java 20

Q) What is the difference between JDK , JRE or JVM ?

JDK

JDK stands for Java Development Kit.

It is a installable software which consist Java Runtime Environment (JRE), Java Virtual Machine (JVM), Compiler (javac) , Interpreter (java), archiever (.jar) , document generator (javadoc) and other tools needed for java application development.

JRE

JRE stands for Java Runtime Environment.

It provides very good environment to run java applications only.

JVM

JVM stands for Java Virtual Machine.

It is used to execute our program line by line procedure.

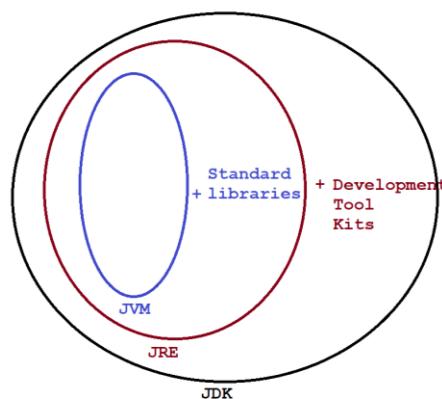


Diagram:java5.1

Identifiers

A name in java is called identifier.

It can be class name, variable name , method name or label name.

ex:

```
class Test
{
    public static void main(String[] args)
    {
```

```

        int i = 10;
        System.out.println(i);
    }
}

```

Here Test,main,args and i are identifiers.

Rules to declare an identifiers

Rule1: Identifier will accept following characters.

ex: A-Z, a-z, 0-9, _, \$

Rule2: If we take other characters then we will get compile time error.

ex: int \$=10; //valid
int _abcd; //valid
int #=20; //invalid

Rule3: Identifier must and should starts with alphabet,underscore or dollar symbol but not with digits.

ex: int a1234; //valid
int _1234; //valid
int 1abcd; //invalid

Rule4: Every identifier is a case sensitive.

ex: int number;
int NUMBER;
int NumBer;

Rule5: We can't take reserved words as an identifier name.

ex: int if;//invalid
int else; //invalid

Rule6: There is no length limit for an identifier but it is not recommended to take more then 15 characters.

Rule7: We can take predefined classes and interfaces as an identifier name but it is not good programming practice.

ex: int System=10;
int Runnable=20;

Reserved words

There are some identifiers which are reserved to associate some functionality or meaning such type of identifiers are called reserved words.

Java supports 53 reserved words and it is divided into two types.

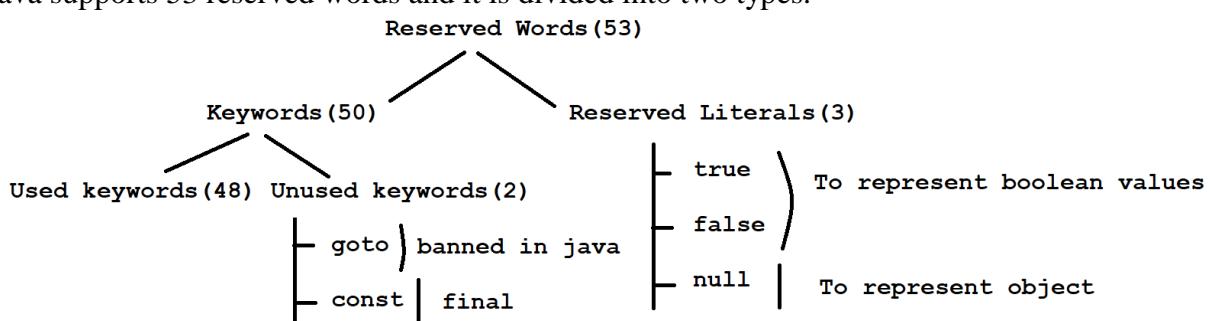


Diagram: java6.1

Used keywords with respect to class

package
import
interface
enum
class
implements
extends

Used keywords with respect to object

new
instanceof
this
super

Used keywords with respect to datatype

byte
short
int
long
float
double
boolean
char

Used keywords with respect to modifiers

default
public
private
protected
final
static
abstract
native
transient
volatile
synchronized
strictfp

Used keywords with respect to flow control

if
else
switch
case
break
continue
do
while
for

Used keywords with respect to return type

void

Used keywords with respect exception handling

try

catch

throw

throws

finally

assert

Java

Version : Java 8

JDK : 1.8v

Creator : James Gosling

Vendor : Oracle Corporation (Sun Micro System)

Open source : Open source

website : www.oracle.com/in/java

Tutorials : www.javatpoint.com

www.javaus.com

www.dzone.com

www.w3school.com

and etc.

Steps to setup Java environmental variables

step1: Make sure JDK 1.8 version installed successfully.

step2: Copy "lib" directory from java_home folder.

ex: C:\Program Files\Java\jdk1.8.0_181\lib

step3: Paste "lib" directory in Environmental variables.

ex:

Right click to MyPC --> properties --> Advanced system settings

Environmental variables -->

User variables --> click to new button -->

variable Name : CLASSPATH

variable value : C:\Program Files\Java\jdk1.8.0_181\lib;

System variables --> click to new button -->

variable Name : path

variable value : C:\Program Files\Java\jdk1.8.0_181\bin;

--> ok ---> ok --->ok.

step4: Check the environmental setup done perfectly or not.

ex: cmd> javap

cmd> java -version

Steps to develop first java application

- step1: Make sure JDK 1.8 installed successfully.
- step2: Make sure Environment setup done perfectly.
- step3: Create a "javaprog" folder inside 'E' drive.
- step4: Open the nodepad and develop simple Hello World program.

ex:

```
class Test
{
    public static void main(String[] args)
    {
        System.out.println("Hello World");
    }
}
```

- step5: Save above java program with same name as class name inside javaprog folder.

- step6: Open the command prompt from javaprog folder or location.

- step7: Compile the java program by using below command.

ex: javaprog> javac Test.java

|
filename

- step8: Run the java program by using below command.

ex: javaprog> java Test

|
classname

Interview Questions

Q)What is package?

A package is a collection of classes and interfaces.

Q)Which package is a default package in java?

java.lang package is a default package in java.

Q)What is Program?

A program is a set of instructions and instruction is a set of tokens.

Q)What is application?

Application is a collection of programs.

Q)What is software?

Software is a collection of applications.

Datatypes

Datatype describes what type of value we want to store inside a variable.

Datatype also tells how much memory has to be created for a variable.

In java datatypes are divided into two types.

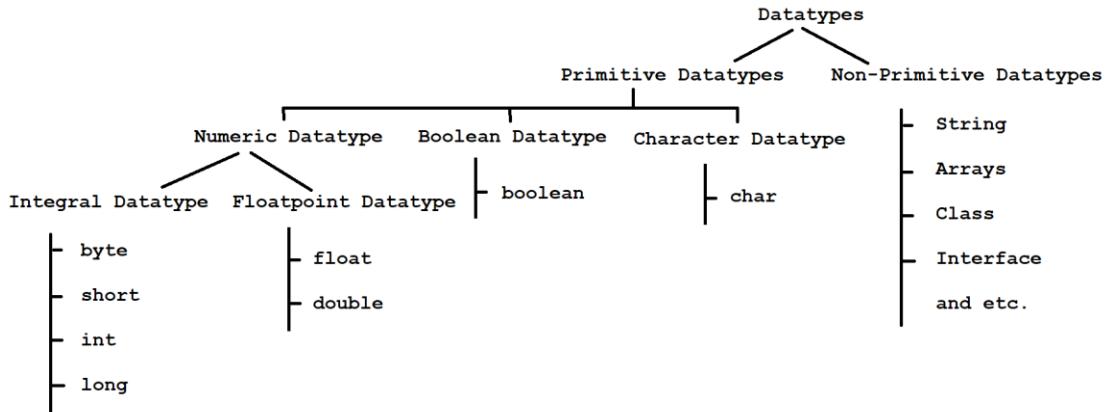


Diagram: java7.1

byte

It is a smallest datatype in java.

Size: 1 byte (8 bits)

Range: -128 to 127 (- 2^7 to 2^7-1)

ex:
 1) byte b=10;
 System.out.println(b); // 10

2) byte b=130;
 System.out.println(b); // C.T.E

3) byte b=10.5;
 System.out.println(b); // C.T.E

short

It is rarely used datatype in java.

Size: 2 bytes (16 bits)

Range : -32768 to 32767 (- 2^{15} to $2^{15}-1$)

ex:
 1) byte b=100;
 short s=b;
 System.out.println(s); // 100

2) short s="hi";
 System.out.println(s); // C.T.E

3) short s=true;
 System.out.println(s); // C.T.E

int

It is mostly used datatype in java.

Size: 4 bytes (32 bits)

Range: -2147483648 to 2147483647 (- 2^{31} to $2^{31}-1$)

ex:
 1) int i=10.5;
 System.out.println(i); // C.T.E

2) int i="false";
 System.out.println(i); // C.T.E

```
3) int i=true;  
System.out.println(i); // C.T.E
```

```
4) int i='a';  
System.out.println(i); // 97
```

Note: In java for every character we have universal unicode value.

ex: A ---- 65
 a ---- 97

long

If int datatype is not enough to hold large value then we need to use long datatype.

Sze: 8 bytes (64 bits)

Range: (-2^63 to 2^63-1)

ex: 1) long l=10.5;
 System.out.println(l); // C.T.E

```
2) long l=true;  
System.out.println(l); // C.T.E
```

```
3) long l="hi";  
System.out.println(l); // C.T.E
```

```
4) long l=10;  
System.out.println(l); //10
```

```
5) long l='A';  
System.out.println(l); // 65
```

float

If we depend upon 4 to 6 decimal point of accuracy then we need to float.

Size: 4 bytes (32 bits)

Range: -3.4e38 to 3.4e38

To represent float value we need to suffix with 'f'.

ex:
 10.5f

ex:

```
1) float f=10.5f;  
System.out.println(f); // 10.5
```

```
2) float f=10;  
System.out.println(f); // 10.0
```

double

If we depend upon 14 to 16 decimal point of accuracy then we need to use double.

Size: 8 bytes (64 bits)

Range: -1.7e308 to 1.7e308

To represent double value we need to suffix with 'd'.

ex:
 10.5d

```
3) float f='a';  
System.out.println(f); //97.0
```

ex:

```
1) double d=10.5d;  
System.out.println(d); // 10.5
```

```
2) double d=10;  
System.out.println(d); // 10.0
```

```
3) double d='A';  
System.out.println(d); //65.0
```

boolean

It is used to represent boolean values either true or false.

Size : (Not Applicable - 1 bit)

Range : (Not Applicable)

ex:

```
1) boolean b="false";  
System.out.println(b); // C.T.E
```

```
2) boolean b=TRUE;  
System.out.println(b); // C.T.E
```

```
3) boolean b=true;  
System.out.println(b) // true
```

char

It will take single character which is enclosed in a single quotation.

Size : 2 bytes (16 bits)

Range : 0 to 65535

ex:

```
1) char c='a';  
System.out.println(c); // a
```

```
2) char c=65;  
System.out.println(c); // A
```

```
3) char c="a";  
System.out.println(c); // C.T.E
```

Datatype	Size	Range	Wrapper class	Default value
byte	1 byte	-128 to 127	Byte	0
short	2 bytes	-32768 to 32767	Short	0
int	4 bytes	-2147483648 to 2147483647	Integer	0
long	8 bytes	-2^63 to 2^63-1	Long	0L
float	4 bytes	-3.4e38 to 3.4e38	Float	0.0
double	8 bytes	-1.7e308 to 1.7e308	Double	0.0
boolean	-	-	Boolean	false
char	2 bytes	0 to 65535	Character	0 (space)

Diagram: java7.2

Q) Write a java console to display the range of byte datatype?

class Test

```
{
    public static void main(String[] args)
    {
        System.out.println(Byte.MIN_VALUE);
        System.out.println(Byte.MAX_VALUE);
    }
}
```

Q) Write a java console to display the range of int datatype?

class Test

```
{
    public static void main(String[] args)
    {
        System.out.println(Integer.MIN_VALUE);
        System.out.println(Integer.MAX_VALUE);
    }
}
```

Q) Is java purely object oriented or not?

No, Java will not consider as purely object oriented programming language because it does not support many OOPS concepts like multiple inheritance, operator overloading and more ever we depends upon primitive datatypes which are non-objects.

Internal Architecture of JVM

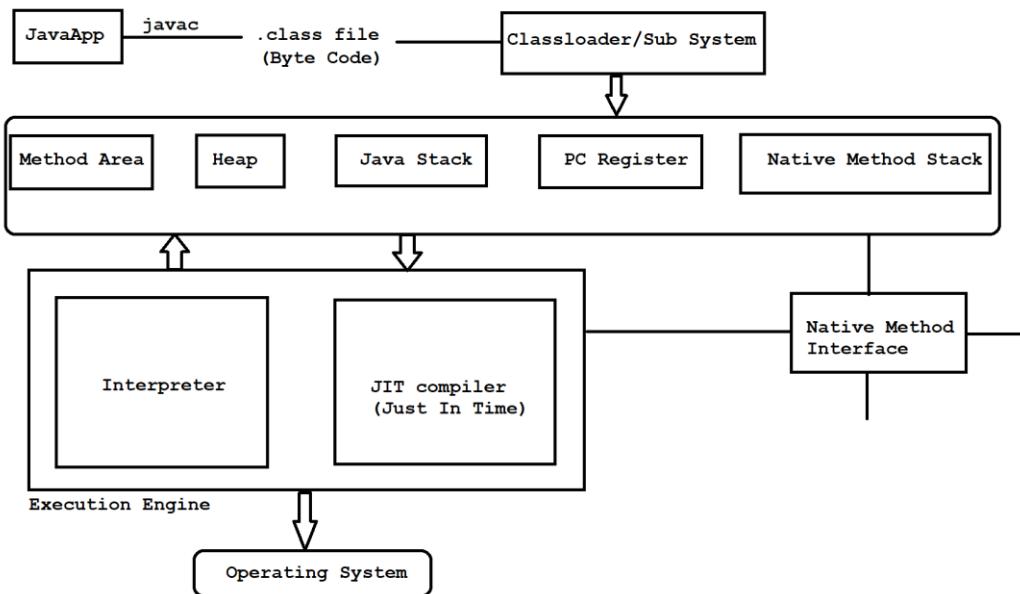


Diagram: java8.1

Our java application contains java code instructions. Once if we compile ,java code instructions convert to byte code instructions in .class file.

JVM will invoke a module called classloader or subsystem to load all the byte code instructions from .class file. The work of classloader is to check byte code instructions are proper or not. If it is not proper then it will refuse the execution. If it is proper then it will allocate the memories.

We have five types of memories.

1)Method Area

Method area contains code of a variable, code of a method and code of a class.

2)Heap

Our object creation will store in heap area.

Note: Whenever JVM loads byte code instructions from .class file it will create method area and heap area automatically.

3)Java Stack

Java methods will store in method area but to execute those methods we required some memory. That memory will be allocated in java stack.

4)PC register

It is a program counter register which is used to track the address of an instructions.

5)Native Method Stack

Java methods will execute in method area. Similary native methods will execute in native method stack.

But we can't execute native methods directly we required a program called Native method interface.

Execution Engine

Execution engine contains interpreter and JIT compiler.

Whenever JVM loads byte code instructions from .class file it will use interpreter and JIT compiler simultaneously.

Interpreter is used to execute our program line by line procedure.

JIT compiler is used to increase the execution speed of our program.

Interview Questions

Q)A .class file contains what code ?

Byte code

Q)How many memories are there in java?

There are five memories in java.

- 1) Method area
- 2) Heap
- 3) Java Stack
- 4) PC register
- 5) Native method stack

Q)What is JIT compiler ?

It is a part of a JVM which is used to increase the execution speed of our program.

Q)How many classloaders are there in java?

We have three predefined classloaders in java.

- 1)Bootstrap classloader (It loads rt.jar file)
- 2)Extension classloader (It loads all the jar file from ext folder)
- 3)Application/System classloader (It loads .class file from CLASSPATH)

Types of variables

A name which is given to a memory location is called variable.

Purpose of variable is used to store the data.

In java, We have two types of variables.

1)Primitive variables

It is used to represent primitive values.

2)Reference variables

It is used to represent object reference.

ex: Student s=new Student();

|

reference variable

Based on the position and execution these variables are divided into three types.

1)Instance variables / Non-static variables

2)Static variables / Global variables

3)Local variables / Temperory variables / Automatic variables

1) Instance variables

A value of a variable which is varied(changes) from object to object is called instance variable.
Instance variable will be created at the time of object creation and it will destroy at the time of object destruction.Hence scope of instance variable is same as scope of an object.

Instance variable store in Heap area as a part of an object.

Instance variable must and should declare immediately after the class but not inside methods, blocks and constructors.

Instance variable we can access directly from instance area but we can't access directly from static area.

To access instance variable from static area we need to create object reference.

ex:1

```
class Test
{
    //instance variable
    int i = 10;
    public static void main(String[] args)
    {
        System.out.println(i); //C.T.E
    }
}
```

ex:2

```
class Test
{
    //instance variable
    int i = 10;
    public static void main(String[] args)
    {
        Test t=new Test();
        System.out.println(t.i); //10
    }
}
```

Note: If we won't initialize any value to instance variable then JVM will initialized default values.

ex:3

```
class Test
{
    //instance variable
    boolean b;
    public static void main(String[] args)
    {
        Test t=new Test();
        System.out.println(t.b); //false
    }
}
```

```

ex:4
class Test
{
    public static void main(String[] args)
    {
        //calling
        Test t=new Test();
        t.m1();
    }
    //non-static method
    public void m1()
    {
        System.out.println("instance-method");
    }
}

```

2)Static variables

A value of a variable which is not varied from object to object is called static variable.
A static variable will be created at the time of class loading and it will destroy at the time of class unloading .Hence scope of static variable is same as scope of .class file.

Static variable will store in method area.

Static variable must and should declare immediately after the class by using static keyword but not inside methdods,blocks and constructors.

Static variable can access directly from instance area and static area.

Static varible can access by using object reference and classname.

ex:1

```

class Test
{
    //static variable
    static int i=10;
    public static void main(String[] args)
    {
        System.out.println(i);//10
        Test t=new Test();
        System.out.println(t.i); //10
        System.out.println(Test.i);//10
    }
}

```

Note: If we won't initialize any value to static variable then JVM will initialized default values.

ex:2

```

class Test
{
    //static variable

```

```

static String s;
public static void main(String[] args)
{
    System.out.println(s);//null
}
}

```

ex:3

```

class Test
{
    public static void main(String[] args)
    {
        //calling
        m1();
        Test t=new Test();
        t.m1();
        Test.m1();
    }
    //static method
    public static void m1()
    {
        System.out.println("static-method");
    }
}

```

3)Local variables

To meet temporary requirement, programmers will declare some variables inside methods , blocks and constructors such type of variables are called local variables.

A local variable will be created at the time of execution block and it will destroy when execution block is executed.Hence scope of local variable is same as scope of execution block where it is declared.

Local variable will store in java stack memory.

ex:

```

class Test
{
    public static void main(String[] args)
    {
        //local variable
        int i=10;
        System.out.println(i);//10
    }
}

```

Note: IF we won't initialize any value to local variable then JVM will not initialized any default value.

```
ex:2
class Test
{
    public static void main(String[] args)
    {
        //local variable
        int i;
        System.out.println(i); //C.T.E
    }
}
```

o/p: variable i might not have been initialized

Main method

Our program contains main method or not.

Either it is properly declared or not.

It is not a responsibility of a compiler to check.

It is a liability of a JVM to look for main method always at runtime.

JVM always look for main method with following signature.

syntax: public static void main(String[] args)

If we perform any changes in above signature then JVM will throw one runtime error called main method not found.

public	JVM wants to call main method from anywhere.
static	JVM wants to call main method without using object reference.
void	Main method does not return anything to JVM.
main	It is a identifier given to a main method.
String[] args	It is a command line argument.

We can perform following changes in main method.

1) Order of modifiers is not important , incase of public static we can declare static public also.

ex: static public void main(String[] args)

2) We change String[] in following acceptable formats.

ex: public static void main(String[] args)

public static void main(String []args)

public static void main(String args[])

3) We can replace String[] with var-arg parameter.

ex: public static void main(String... args)

4) We can change args with any java valid identifier.

ex: public static void main(String[] ihub)

5) Main method will accept following modifiers.

ex: synchronized

strictfp

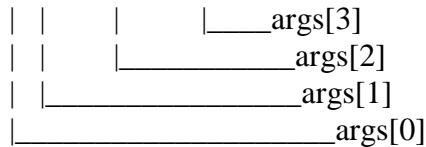
final

Command Line arguments

Arguments which are passing through command prompt such type of arguments are called command line arguments.

In command line arguments we need to pass out input values always at runtime.

ex: javac Test.java
 java Test 101 raja M 1000.0



ex:

```
class Test
{
    public static void main(String[] args)
    {
        System.out.println(args[0]);
        System.out.println(args[1]);
        System.out.println(args[2]);
        System.out.println(args[3]);
    }
}
```

System.out.println()

It is a output statement in java.

Whenever we want to display any data or userdefined statements then we need to use output statement.

syntax: static variable

```
|
System.out.println(" ");
|
|           |
predefined   predefined method
final class
```

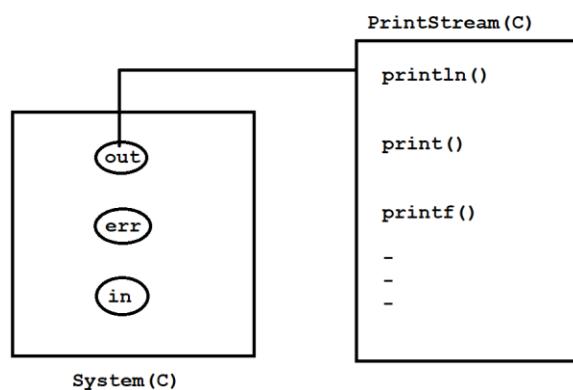


Diagram: java9.1

Various ways to display the data

- 1) System.out.println("Hello World");
- 2) int i=10;
 System.out.println(i);
 System.out.println("The value is =" + i);
- 3) int i=10,j=20;

```
System.out.println(i+" "+j);
System.out.println(i+" and "+j);
4) int i=1,j=2,k=3;
System.out.println(i+" "+j+" "+k);
```

Q) What is the difference b/w System.out.println() and System.err.println()?

System.out.println()

It is used to display the output only one console.

ex:

```
class Test
{
    public static void main(String[] args)
    {
        System.out.println("Hello World");
    }
}
```

System.err.println()

It is used to display the output on console as well it will redirect to some physical file.

ex:

```
class Test
{
    public static void main(String[] args)
    {
        System.err.println("Hello World");
    }
}
```

o/p: javac Test.java
java Test 2>abc.txt

Fully Qualified Name

Fully qualified name means we will declare our class or interface along with package name.

It is used to improve the readability of our code.

ex:

```
class Test
{
    public static void main(String[] args)
    {
        java.util.Date d=new java.util.Date();
        System.out.println(d);
    }
}
```

Import Statements

Whenever we use import statement we should not fully qualified name.

Using short name also we can achieve.

We have three types of import statements in java.

- 1)Explicit class import
- 2)Implicit class import
- 3)static import

1)Explicit class import

This type of import statement is highly recommended to use because it will improve readability of our code.

ex:

```
import java.time.LocalDate;
import java.time.LocalTime;
class Test
{
    public static void main(String[] args)
    {
        LocalDate date=LocalDate.now();
        System.out.println(date);
        LocalTime time=LocalTime.now();
        System.out.println(time);
    }
}
```

2)Implicit class import

This type of import statement is not recommended to use because it will reduce the readability of our code.

ex:

```
import java.time.*;
class Test
{
    public static void main(String[] args)
    {
        LocalDate date=LocalDate.now();
        System.out.println(date);
        LocalTime time=LocalTime.now();
        System.out.println(time);
    }
}
```

3)static import

Using static import we can call static members directly.

Often use of static makes our program unreadable or complex.

ex:

```
import static java.lang.System.*;
class Test
{
    public static void main(String[] args)
    {
        out.println("stmt1");
        out.println("stmt2");
    }
}
```

```

        out.println("stmt3");
    }
}

ex:
import static java.lang.System.*;
class Test
{
    public static void main(String[] args)
    {
        out.println("stmt1");
        exit(0);
        out.println("stmt3");
    }
}

```

Basic Java Programs

Q) Write a java program to display sum of two numbers?

```

import java.util.Scanner;
class Example1
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the First Number :");
        int a=sc.nextInt();
        System.out.println("Enter the Second Number :");
        int b=sc.nextInt();
        //logic
        int c=a+b;
        System.out.println("sum of two numbers is =" +c);
    }
}

```

Q) Write a java console to perform sum of two numbers without using third variable?

```

import java.util.Scanner;
class Example2
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the First Number :");
        int a=sc.nextInt();
        System.out.println("Enter the Second Number :");
        int b=sc.nextInt();
        System.out.println("sum of two numbers is =" +(a+b));
    }
}

```

```
    }  
}
```

Q)Write a java program to find out square of a given number?

input: 5

output: 25

```
import java.util.Scanner;  
class Example3  
{  
    public static void main(String[] args)  
    {  
        Scanner sc=new Scanner(System.in);  
        System.out.println("Enter the Number :");  
        int n=sc.nextInt();  
        int square=n*n;  
        System.out.println("square of a given number is =" +square);  
    }  
}
```

Q)Write a java program to find out cube of a given number?

input: 5

output:125

ex:

```
import java.util.Scanner;  
class Example4  
{  
    public static void main(String[] args)  
    {  
        Scanner sc=new Scanner(System.in);  
        System.out.println("Enter the Number :");  
        int n=sc.nextInt();  
        int cube=n*n*n;  
        System.out.println("Cube of a given number is =" +cube);  
    }  
}
```

Q)Write a java program to find out area of a circle?

import java.util.Scanner;

class Example5

```
{  
    public static void main(String[] args)  
    {  
        Scanner sc=new Scanner(System.in);  
        System.out.println("Enter the radius :");  
    }  
}
```

```

        int r=sc.nextInt();
        //logic
        float area=3.14f*r*r;
        System.out.println("Area of a circle is =" +area);
    }
}

```

Q)Write a java program to find out perimeter of a circle?

```

import java.util.Scanner;
class Example6
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the radius :");
        int r=sc.nextInt();
        //logic
        float perimeter=2*3.14f*r;
        System.out.println("Perimeter of a circle is =" +perimeter);
    }
}

```

Q)Write a java program to perform swapping of two numbers?

```

input: a = 10 and b = 20
output: a = 20 and b = 10
import java.util.Scanner;
class Example7
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the First Number :");
        int a=sc.nextInt();
        System.out.println("Enter the Second Number :");
        int b=sc.nextInt();

        System.out.println("Before swapping a=" +a+ " and b=" +b);
        //logic
        int temp=a;
        a=b;
        b=temp;
        System.out.println("After swapping a=" +a+ " and b=" +b);

    }
}

```

Q)Write a java program to perform swapping of two numbers without using third variable?

```
import java.util.Scanner;
class Example8
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the First Number :");
        int a=sc.nextInt();
        System.out.println("Enter the Second Number :");
        int b=sc.nextInt();
        System.out.println("Before swapping a="+a+" and b="+b);
        //logic
        a=a+b;
        b=a-b;
        a=a-b;
        System.out.println("After swapping a="+a+" and b="+b);
    }
}
```

Q)What is a java program to accept salary and find out 10% of TDS ?

```
import java.util.Scanner;
class Example9
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the salary :");
        int salary=sc.nextInt();
        //logic
        float tds=(float)salary*10/100;
        System.out.println("10 percent of TDS is ="+tds);

    }
}
```

Q)Write a java program to convert CGPA to percentage?

```
import java.util.Scanner;
class Example10
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the CGPA :");
```

```

        float cgpa=sc.nextFloat();

        //logic
        float percentage=cgpa*9.5f;
        System.out.println("CGPA to percentage is =" +percentage);
    }
}

```

Assignment

- Q) Write a java program to find out area of a rectangle?
 Q) Write a java program to find out area of a triangle?

Typecasting

The process of converting from one datatype to another datatype is called typecasting.
 In java, typecasting can be performed in two ways.

- 1) Implicit typecasting
- 2) Explicit typecasting

1) Implicit typecasting

If we want to store small value into a bigger variable then we need to use implicit typecasting.
 A compiler is responsible to perform implicit typecasting.

There is no possibility to loss the information.

It is also known as widening or upcasting.

We can perform implicit typecasting as follow.

ex: byte ---> short
 --->
 int -->long -->float -->double
 --->
 char

ex:

```

class Test
{
    public static void main(String[] args)
    {
        char ch='a';

        long l=ch;

        System.out.println(l); //97
    }
}
```

ex:

```

class Test
{
    public static void main(String[] args)
```

```

{
    int i=10;

    double d=i;

    System.out.println(d); // 10.0
}
}

```

ex:

```

class Test
{
    public static void main(String[] args)
    {
        byte b=10;

        int i=b;

        System.out.println(i); // 10
    }
}

```

2)Explicit typecasting

If we want store big value into a smaller variable then we need to use explicit typecasting.

A programmer is responsible to perform explicit typecasting.

There is a possibility to loss the information.

It is also known as Narrowing or Downcasting.

We can perform explicit typecasting as follow.

ex: byte <--- short

```

<---
      int <-- long <--float <--double
<---
      char

```

ex:

```

class Test
{
    public static void main(String[] args)
    {
        int i=65;

        char ch=(char)i;

        System.out.println(ch);//A
    }
}

```

```
ex:  
class Test  
{  
    public static void main(String[] args)  
    {  
        float f=10.5f;  
  
        int i=(int)f;  
  
        System.out.println(i);//10  
    }  
}
```

```
ex:  
class Test  
{  
    public static void main(String[] args)  
    {  
        int i=130;  
  
        byte b=(byte)i;  
  
        System.out.println(b); // -126  
    }  
}
```

Java Source File Structure

case1:

A java program can have multiple classes.

If a java program contains multiple classes then we need to check which class contains main method , that class will consider as main class.

```
ex: B.java  
class A  
{  
    -  
}  
class B  
{  
    public static void main(String[] args)  
    {  
        -  
    }  
}
```

If we compile above program we will get two .class files i.e A.class and B.class.

case2:

If a java program contains multiple classes with main method then we need to declare one class as public and that class will consider as main class.

ex:

```
A.java
-----
public class A
{
    public static void main(String[] args)
    {
        System.out.println("A class");
    }
}
class B
{
    public static void main(String[] args)
    {
        System.out.println("B class");
    }
}
class C
{
    public static void main(String[] args)
    {
        System.out.println("C class");
    }
}
>javac A.java (A.class,B.class and C.class)
>java A (A class will execute)
>java B (B class will execute)
>java C (C class will execute )
```

Types of Blocks in java

A block is a set of statements which is enclosed in a curly braces i.e {}.

syntax:

```
//block
{
    -
    - //set of statements
    -
}
```

We have three types of blocks in java.

- 1)Instance block
- 2)Static block
- 3)Local block

1)Instance block

Instance block is used to initialize the instance variable.

Instane block will execute when we create an object.

We can declare instance block as follow.

syntax:

```
//instance block
{
    -
    - //set of statements
    -
}
```

Instance block must and should declare immediately after the class but not inside methods and constructors.

ex:1

```
class Test
{
    //instance block
    {
        System.out.println("instance-block");
    }
}
```

```
public static void main(String[] args)
{
    System.out.println("main-method");
}
```

o/p: main-method

ex:

```
class Test
{
    //instance block
    {
        System.out.println("instance-block");
    }
}
```

```
public static void main(String[] args)
{
    System.out.println("main-method");
    Test t=new Test();
}
```

o/p: main-method
instance-block

ex:

```
class Test
{
    //instance block
    {
        System.out.println("instance-block");
    }

    public static void main(String[] args)
    {
        Test t1=new Test();
        System.out.println("main-method");
        Test t2=new Test();
    }
}
```

o/p: instance-block
main-method
instance-block

ex:

```
class Test
{
    //instance variable
    int i;

    //instance block
    {
        i=100;
    }

    public static void main(String[] args)
    {
        Test t=new Test();
        System.out.println(t.i);//100
    }
}
```

2)Static block

Static block is used to initialize the static variables.
Static block will execute at the time of classloading.
We can declare static block as follow.

syntax:

```
//static block
static
{
```

- //set of statements

-

}

Static block must and should declare immediate after the class using static keyword but not inside methods and constructors.

ex:1

```
class Test
{
    //static block
    static
    {
        System.out.println("static-block");
    }
```

```
    public static void main(String[] args)
    {
        System.out.println("main-method");
    }
```

}

o/p: static-block
main-method

ex:2

```
class Test
{
    //static block
    static
    {
        System.out.println("static-block");
    }

    //instance block
    {
        System.out.println("instance-block");
    }
```

```
    public static void main(String[] args)
    {
        Test t=new Test();
        System.out.println("main-method");
    }
```

}

o/p: static-block
instance-block
main-method

```

ex:3
class Test
{
    //static variable
    static int i;

    //static block
    static
    {
        i=200;
    }

    public static void main(String[] args)
    {
        System.out.println(i);//200
    }
}

```

3)Local block

A local block is used to initialize the local variables.

Local will execute just like normal statement.

We can declare local block as follow.

syntax:

```

//local block
{
    -
    - //set of statements
    -
}

```

Local block must and should declare inside methods and constructors.

ex:1

```

class Test
{
    public static void main(String[] args)
    {
        System.out.println("stmt1");
        //local block
        {
            System.out.println("stmt2");
        }
        System.out.println("stmt3");
    }
}

```

o/p: stmt1
stmt2
stmt3

```
ex:2
class Test
{
    public static void main(String[] args)
    {
        //local variable
        int i;

        //local block
        {
            i=300;
        }

        System.out.println(i); // 300
    }
}
```

Interview Question

Q)Can we execute java program without main method ?

Yes , Till 1.6 version it is possible to execute java program without main method by using static block.But from 1.7 version onwards it is not possible to execute java program without main method.

ex:

```
class Test
{
    static
    {
        System.out.println("Hello World");
        System.exit(0);
    }
}
```

Operators

Operator is a symbol which is used to perform some operations on operands.

ex: a + b

Here + is a operator

Here a and b are operands

It can be arithmetic operation, logical operation, bitwise operation , conditional operation and etc.

We have following list of operators in java.

- 1)Assignment operators
- 2)Ternary operators/Conditional operators
- 3)Logical operators
- 4)Bitwise operators
- 5)Arithmetic operators
- 6)Relational operators

7) Unary operators

1) Assignment operators

ex:

```
class Test
{
    public static void main(String[] args)
    {
        int i=10;
        i=20;
        i=30;
        System.out.println(i); // 30
    }
}
```

Note: Reinitialization is possible in java.

ex:

```
class Test
{
    public static void main(String[] args)
    {
        final int i=10;
        i=20;
        i=30;
        System.out.println(i); // C.T.E
    }
}
```

Note: We can't assign the values to final variable

ex:

```
class Test
{
    public static void main(String[] args)
    {
        int i=1,2,3,4,5;
        System.out.println(i); //C.T.E
    }
}
```

Note: Illegal start of expression

ex:

```
class Test
{
    //global variable
```

```

static int i=100;

public static void main(String[] args)
{
    //local variable
    int i=200;

    System.out.println(i); //200
}
}

```

Note: Here priority goes to local variable.

ex:

```

class Test
{
    public static void main(String[] args)
    {
        System.out.println(10%2); // 0
        System.out.println(10%20); // 10
        System.out.println(40%3); // 1
        System.out.println(40%80); // 40
    }
}

```

ex:

```

class Test
{
    public static void main(String[] args)
    {
        System.out.println(10/2); // 5
        System.out.println(10/20); // 0
        System.out.println(40/3); // 13
        System.out.println(40/80); // 0
    }
}

```

2) Ternary operator / Conditional operator

syntax: (condition)?value1:value2;

ex:

```

class Test
{
    public static void main(String[] args)
    {
        boolean b=(5>2)?true:false;
        System.out.println(b); //true
    }
}

```

```
        }
    }
```

ex:

```
class Test
{
    public static void main(String[] args)
    {
        boolean b=(5>20)?true:false;
        System.out.println(b);//false
    }
}
```

ex:

```
class Test
{
    public static void main(String[] args)
    {
        int i=(4>2)?1:0;

        System.out.println(i);//1
    }
}
```

Q)Write a java program to find out greatest of two numbers?

```
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the First Number :");
        int a=sc.nextInt();

        System.out.println("Enter the Second Number :");
        int b=sc.nextInt();

        //logic
        int max=(a>b)?a:b;

        System.out.println("Greatest of two numbers is =" +max);
    }
}
```

```
    }  
}
```

Q)Write a java program to find out greatest of three numbers?

```
import java.util.Scanner;  
class Test  
{  
    public static void main(String[] args)  
    {  
        Scanner sc=new Scanner(System.in);  
  
        System.out.println("Enter the First Number :");  
        int a=sc.nextInt();  
  
        System.out.println("Enter the Second Number :");  
        int b=sc.nextInt();  
  
        System.out.println("Enter the Third Number :");  
        int c=sc.nextInt();  
  
        //logic  
        int max=(a>b)?((a>c)?a:c):((b>c)?b:c);  
  
        System.out.println("Greatest of three numbers is =" +max);  
    }  
}
```

3)Logical operators

logical AND operator(&&)

Truth table

T	T	= T
T	F	= F
F	T	= F
F	F	= F

ex:

```
class Test  
{  
    public static void main(String[] args)  
    {  
        boolean b=((5>6) && (9<10))?true:false;  
        System.out.println(b);//false  
    }  
}
```

```
}
```

ex:

```
class Test
{
    public static void main(String[] args)
    {
        boolean b=((5>2) && (9<10))?true:false;
        System.out.println(b);//true
    }
}
```

logical OR operator(||)

Truth table

T	T	= T
T	F	= T
F	T	= T
F	F	= F

ex:

```
class Test
{
    public static void main(String[] args)
    {
        boolean b=(5>2) || (6<2);

        System.out.println(b);//true
    }
}
```

ex:

```
class Test
{
    public static void main(String[] args)
    {
        boolean b= (5>7) || false;
        System.out.println(b);//false
    }
}
```

```

ex:
---
class Test
{
    public static void main(String[] args)
    {
        boolean b= (5>20) && (9<10) || false;

        System.out.println(b);//false
    }
}

```

Logical NOT operator (!)

```

ex:
class Test
{
    public static void main(String[] args)
    {
        boolean b=!(5>2);

        System.out.println(b);//false
    }
}

```

```

ex:
class Test
{
    public static void main(String[] args)
    {
        boolean b=!(6>90);

        System.out.println(b);//true
    }
}

```

How to convert decimal number to binary number

decimal number : 10

binary number : 1010

$$\begin{array}{r}
 2|10-- 0 \\
 2|5-- 1 \\
 2|2-- 0 \quad ^ \\
 \quad \quad \quad | \\
 \hline
 \quad \quad \quad \\
 1010
 \end{array}$$

How to convert binary number to decimal number
binary number : 1010
decimal number : 10

$$\begin{array}{r} 1010 \\ \text{---} \\ 0*1 + 1*2 + 0*4 + 1*8 \\ 0 + 2 + 0 + 8 \\ 10 \end{array}$$

4) Bitwise Operators

Bitwise AND operator(&)

Bitwise AND operator deals with binary numbers.

Truth table

T	T	= T
T	F	= F
F	T	= F
F	F	= F

ex:

```
class Test
{
    public static void main(String[] args)
    {
        int a=10,b=15;
        int c = a & b;
        System.out.println(c);//10
    }
}
```

10 - 1010
15 - 1111

& - 1010

<--

$$0*1 + 1*2 + 0*4 + 1*8$$

$$0 + 2 + 0 + 8 = 10$$

*/

ex:

```
class Test
{
```

```

public static void main(String[] args)
{
    int a=10,b=5;
    int c = a & b;
    System.out.println(c);//0
}
/*
10 - 1010
5 - 0101
-----
& - 0000
*/

```

Bitwise OR operator(|)

Bitwise OR operator deals with binary numbers.

Truth table

T	T	= T
T	F	= T
F	T	= T
F	F	= F

ex:

```

class Test
{
    public static void main(String[] args)
    {
        int a=10,b=15;
        int c = a | b;
        System.out.println(c);//15
    }
}
/*
10 - 1010
15 - 1111
-----
| - 1111
*/

```

Bitwise XOR operator(^)

Bitwise XOR operator deals with binary numbers.

Truth table

T	T	= F
T	F	= T
F	T	= T
F	F	= F

ex:

```
class Test
{
    public static void main(String[] args)
    {
        int a=10,b=15;
        int c = a ^ b;
        System.out.println(c); //5
    }
}
/*
10 - 1010
15 - 1111
-----
^ - 0101
```

Bitwise NOT operator (\sim)

ex:

```
class Test
{
    public static void main(String[] args)
    {
        int i =  $\sim$ 10;
        System.out.println(i); // -11
    }
}
```

ex:

```
class Test
{
    public static void main(String[] args)
    {
        int i =  $\sim$ 56;
        System.out.println(i); // -57
    }
}
```

ex:

```
class Test
{
```

```

public static void main(String[] args)
{
    int i = ~(-35);

        System.out.println(i); // 34
}

```

5) Arithmetic Operators

% - modules
 / - division
 * - multiplication
 + - addition
 - - subtraction

ex:

```

class Test
{
    public static void main(String[] args)
    {
        int i= 8+6%10+6/3+5*2+7/10+9-12;

        System.out.println(i); //23
    }
}

```

```

/*
8 + 6%10 + 6/3 + 5*2 + 7/10 + 9-12

```

$8 + 6 + 2 + 10 + 0 + 9 - 12$

$35 - 12$

23
*/

6) Relational operators

```

class Test
{
    public static void main(String[] args)
    {
        System.out.println(5>10); // false
        System.out.println(5>=10); // false

        System.out.println(5<10); // true
        System.out.println(5<=5); // true
    }
}

```

```
        System.out.println(10 == 10); // true
        System.out.println(10 == 20); // false
        System.out.println(10 != 10); // false
        System.out.println(10 != 20); // true
    }
}
```

Right Shift operator (>>)

```
10 >> 1 = 10 / 2
10 >> 2 = 10 / 4
10 >> 3 = 10 / 8
10 >> 4 = 10 / 16
10 >> 5 = 10 / 32
```

ex:

```
class Test
{
    public static void main(String[] args)
    {
        int i = 10 >> 3;

        System.out.println(i); // 10 / 8 = 1
    }
}
```

ex:

```
class Test
{
    public static void main(String[] args)
    {
        int i = 100 >> 6;

        System.out.println(i); // 100 / 64 = 1
    }
}
```

Left Shift operator (<<)

```
10 << 1 = 10 * 2
10 << 2 = 10 * 4
10 << 3 = 10 * 8
10 << 4 = 10 * 16
10 << 5 = 10 * 32
```

```
ex:  
class Test  
{  
    public static void main(String[] args)  
    {  
        int i = 10 << 4;  
  
        System.out.println(i); // 10 * 16 = 160  
    }  
}
```

```
ex:  
---  
class Test  
{  
    public static void main(String[] args)  
    {  
        int i = 100 << 2;  
  
        System.out.println(i); // 100 * 4 = 400  
    }  
}
```

Increment/Decrement Operators (++/--)

We have two increment operators.

i) post increment

ex: i++

ii) pre increment

ex: ++i

We have two decrement operators.

i) post decrement

ex: i--

ii) pre decrement

ex: --i

Post increment/decrement

Rule1: First Take

Rule2: Then Change

```
ex:  
class Test  
{  
    public static void main(String[] args)  
    {  
        int i=10;  
        i++;
```

```
        System.out.println(i);//11
    }
}
```

ex:

```
class Test
{
    public static void main(String[] args)
    {
        int i=10;

        System.out.println(i++);//10
    }
}
```

ex:

```
class Test
{
    public static void main(String[] args)
    {
        int i=10;

        int j=i++;

        System.out.println(i+" "+j);//11 10
    }
}
```

ex:

```
class Test
{
    public static void main(String[] args)
    {
        int i=10;

        int j=i-- + i--; // 10 + 9

        System.out.println(i+" "+j);// 8 19
    }
}
```

ex:

```
class Test
{
    public static void main(String[] args)
    {
```

```

        int i=10;

        int j=i++ + i++ + i++; // 10 + 11 + 12
        System.out.println(i+" "+j); // 13 33
    }
}

```

Pre increment/decrement

Rule1: First Change

Rule2: Then Take

ex:

```

class Test
{
    public static void main(String[] args)
    {
        int i=10;

        ++i;

        System.out.println(i); // 11
    }
}

```

ex:

```

class Test
{
    public static void main(String[] args)
    {
        int i=10;

        System.out.println(++i); // 11
    }
}

```

ex:

```

class Test
{
    public static void main(String[] args)
    {
        int i=10;
        int j=++i;
        System.out.println(i+" "+j); // 11 11
    }
}

```

```
ex:  
class Test  
{  
    public static void main(String[] args)  
    {  
        int i=10;  
  
        System.out.println(i++ + ++i);//10 + 12 =22  
    }  
}
```

```
ex:  
class Test  
{  
    public static void main(String[] args)  
    {  
        int i=100;  
  
        100++;  
  
        System.out.println(i);//C.T.E  
    }  
}
```

```
ex:  
class Test  
{  
    public static void main(String[] args)  
    {  
        int i=10;  
  
        System.out.println(++(i++));//C.T.E  
    }  
}
```

Control Statements

Control statement enables the programmer to control flow of the program.
Control statement allows us to make decisions, to jump from one section of code to another section and to execute the code repeatedly.

We have four control statements in java.

- 1)Decision making statement
- 2)Selection statement
- 3)Iteration statement
- 4)Jump statement

1) Decision making statement

It is used to create a conditions in our program.

Decision making statement is possible by using following ways.

- i) if stmt
- ii) if else stmt
- iii) if else if ladder
- iv) nested if stmt

i) if stmt

It will execute the source code only if our condition is true.

syntax:

```
if(condition/expression)
{
    -
    - //code to be execute
    -
}
```

ex:

```
-----
class Test
{
    public static void main(String[] args)
    {
        System.out.println("stmt1");
        if(!(5>2))
        {
            System.out.println("stmt2");
        }
        System.out.println("stmt3");
    }
}
```

o/p:

```
stmt1
stmt2
stmt3
```

ex:

```
-----
class Test
{
    public static void main(String[] args)
    {
        System.out.println("stmt1");
        if(!(5>2))
        {
            System.out.println("stmt2");
        }
    }
}
```

```
        }
    System.out.println("stmt3");
}
}

o/p:
```

```
stmt1
stmt3
```

ex:

```
class Test
{
    public static void main(String[] args)
    {
        if((5>2) && (8<3))
            System.out.println("stmt1");
            System.out.println("stmt2");
            System.out.println("stmt3");
    }
}
```

Note:

Declaration of curly braces is optional.

If we won't defined curly braces then compiler will add curly braces at the time of compilation only to first statement.

Q)Write a java program to find out greatest of two numbers?

```
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the First Number :");
        int a=sc.nextInt();
        System.out.println("Enter the Second Number :");
        int b=sc.nextInt();

        if(a>b)
            System.out.println(a+" is greatest");
        if(b>a)
            System.out.println(b+" is greatest");

    }
}
```

Q)Write a java program to find out greatest of three numbers?

```
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the First Number :");
        int a=sc.nextInt();

        System.out.println("Enter the Second Number :");
        int b=sc.nextInt();

        System.out.println("Enter the Third Number :");
        int c=sc.nextInt();

        if((a>b) && (a>c))
            System.out.println(a+" is greatest");
        if((b>a) && (b>c))
            System.out.println(b+" is greatest");
        if((c>a) && (c>b))
            System.out.println(c+" is greatest");

    }
}
```

ii)if else stmt

It will execute the source code either our condition is true or false.

syntax:

```
if(condition/expression)
{
    - //code to be execute if cond is true
}
else
{
    - //code to be execute if cond is false
}
```

ex:

```
class Test
{
    public static void main(String[] args)
    {
        System.out.println("stmt1");
```

```
if(true)
{
    System.out.println("stmt2");
}
else
{
    System.out.println("stmt3");
}
System.out.println("stmt4");
}

o/p:
stmt1
stmt2
stmt4
```

ex:

```
class Test
{
    public static void main(String[] args)
    {
        System.out.println("stmt1");
        if(false)
        {
            System.out.println("stmt2");
        }
        else
        {
            System.out.println("stmt3");
        }
        System.out.println("stmt4");
    }
}

o/p:
stmt1
stmt3
stmt4
```

Q)Write a java program to find out given age is eligible to vote or not?

```
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
```

```

        System.out.println("Enter the age :");
        int age=sc.nextInt();

        if(age>=18)
            System.out.println("U r eligible to vote");
        else
            System.out.println("U r not eligible to vote");
    }
}

```

Q)Write a java program to check given number is even or odd?

```

import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the number :");
        int n=sc.nextInt();

        if(n%2==0)
            System.out.println("It is even number");
        else
            System.out.println("It is odd number ");
    }
}

```

Q)Write a java program to check given number is odd or not ?

```

import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the number :");
        int n=sc.nextInt();

        if(n%2==1 || n%2!=0)
            System.out.println("It is odd number");
        else
            System.out.println("It is not odd number ");
    }
}

```

Q)Write a java program to find out given year is a leap year or not?

```
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the year :");
        int year=sc.nextInt();

        if(year%4==0)
            System.out.println("It is a leap year");
        else
            System.out.println("It is not a leap year ");
    }
}
```

Q)Write a java program to find out given number is +ve or -ve ?

```
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the number :");
        int n=sc.nextInt();

        if(n==0)
        {
            System.out.println("It is not a positive or negative number");
            System.exit(0);
        }

        if(n>0)
            System.out.println("It is a positive number");
        else
            System.out.println("It is a negative number ");
    }
}
```

iii) if else if ladder

It will execute the source code based on multiple conditions.

syntax:

```
if(cond1)
```

```

{
    - //code to be execute if cond1 is true
}
else if(cond2)
{
    - //code to be execute if cond2 is true
}
else if(cond3)
{
    - //code to be execute if cond3 is true
}
else
{
    - //code to be execute if all conditions are false.
}

```

ex:

```

import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the option :");
        int opt=sc.nextInt();

        if(opt==100)
            System.out.println("It is a police number");
        else if(opt==103)
            System.out.println("It is a enquiry number");
        else if(opt==108)
            System.out.println("It is a emergency number");
        else
            System.out.println("Invalid option");
    }
}

```

Q)Write a java program to find out given alphabet is a vowel or not?

```

import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

```

```

        System.out.println("Enter the alphabet :");
        char ch=sc.next().charAt(0);

        if(ch=='a')
            System.out.println("It is a vowel");
        else if(ch=='e')
            System.out.println("It is a vowel");
        else if(ch=='i')
            System.out.println("It is a vowel");
        else if(ch=='o')
            System.out.println("It is a vowel");
        else if(ch=='u')
            System.out.println("It is a vowel");
        else
            System.out.println("It is not a vowel");
    }
}

```

Q)Write a java program to find out given alphabet is a upper case letter, lower case letter, digit or a special symbol?

```

import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the alphabet :");
        char ch=sc.next().charAt(0);

        if(ch>='a' && ch<='z')
            System.out.println("It is a lowercase letter");
        else if(ch>='A' && ch<='Z')
            System.out.println("It is a uppercase letter");
        else if(ch>='0' && ch<='9')
            System.out.println("It is a Digit");
        else
            System.out.println("It is a Special symbol");
    }
}

```

iv)nested if stmt

If stmt contains another if stmt is called nested if stmt.

syntax:

```
if(condition)
```

```
{  
    if(condition)  
    {  
        - //code to be execute  
        -  
    }  
}
```

ex:1

```
class Test  
{  
    public static void main(String[] args)  
    {  
        System.out.println("stmt1");  
        if(true)  
        {  
            System.out.println("stmt2");  
            if(2>1)  
            {  
                System.out.println("stmt3");  
            }  
            System.out.println("stmt4");  
        }  
        System.out.println("stmt5");  
    }  
}
```

o/p:

```
stmt1  
stmt2  
stmt3  
stmt4  
stmt5
```

ex:

```
class Test  
{  
    public static void main(String[] args)  
    {  
        System.out.println("stmt1");  
        if(false)  
        {  
            System.out.println("stmt2");  
            if(2>1)  
            {  
            }  
        }  
    }  
}
```

```
        System.out.println("stmt3");
    }
    System.out.println("stmt4");
}
System.out.println("stmt5");
}
```

o/p:

```
stmt1
stmt5
```

ex:

```
class Test
```

```
{
```

```
    public static void main(String[] args)
    {
        System.out.println("stmt1");
        if(true)
        {
            System.out.println("stmt2");
            if(2>10)
            {
                System.out.println("stmt3");
            }
            System.out.println("stmt4");
        }
        System.out.println("stmt5");
    }
}
```

o/p:

```
stmt1
stmt2
stmt4
stmt5
```

Q)Write a java program to find out given number is +ve or -ve by using nested if stmt?

```
import java.util.Scanner;
```

```
class Test
```

```
{
```

```
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
    }
}
```

```
System.out.println("Enter the number :");
int n=sc.nextInt();
```

```

        if(n!=0)
        {
            if(n>0)
            {
                System.out.println("It is +ve number");
                System.exit(0);
            }
            System.out.println("It is -ve number");
        }
    }
}

```

2)Selection statement

Switch case

It will execute the source code based on multiple conditions.

It is similar to if else if ladder.

syntax:

```

switch(condition/expression)
{
    case val1:
        //code to be execute
        break stmt;
    case val2:
        //code to be execute
        break stmt;
    -
    -
    default:
        //code to be execute if all cases are false.
}

```

ex:

```

import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the option :");
        int opt=sc.nextInt();

        switch(opt)
        {

```

```

        case 100: System.out.println("It is police number ");
                    break;
        case 103: System.out.println("It is enquiry number");
                    break;
        case 108: System.out.println("It is emergency number");
                    break;
        default: System.out.println("Invalid option");
    }

}

```

Declaration of break statement in switch case is optional.

If we won't defined break statement then from where our condition is satisfied from there all cases will be executed that state is called fall through state of switch case.

ex:

```

import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the option :");
        int opt=sc.nextInt();

        switch(opt)
        {
            case 100: System.out.println("It is police number ");
                        //break;
            case 103: System.out.println("It is enquiry number");
                        //break;
            case 108: System.out.println("It is emergency number");
                        //break;
            default: System.out.println("Invalid option");
        }
    }
}

```

The allowed datatype of switch case are byte,short,int ,char and String.
If we take other datatypes then we will get compile time error.

```

ex:
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the String :");
        String str=sc.next();

        switch(str)
        {
            case "one": System.out.println("January"); break;
            case "two": System.out.println("February"); break;
            case "three": System.out.println("March"); break;
            case "four": System.out.println("April"); break;
            case "five": System.out.println("May"); break;
            default: System.out.println("Coming Soon...");
        }
    }
}

```

Q)Write a java program to check given alphabet is a vowel or consonent?

ex:

```

import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the Alphabet :");
        char ch=sc.next().charAt(0);
        switch(ch)
        {
            case 'a': System.out.println("It is a vowel"); break;
            case 'e': System.out.println("It is a vowel"); break;
            case 'i': System.out.println("It is a vowel"); break;
            case 'o': System.out.println("It is a vowel"); break;
            case 'u': System.out.println("It is a vowel"); break;
            default: System.out.println("It is a consonent");
        }
    }
}

```

Assignment

Q) Write a java program to accept six marks of a student then find out total, average and grade?
i) if average is greater than equals to 75 then A grade.
ii) if average is greater than equals to 50 then B grade.
iii) if average is greater than equals to 35 then C grade.
iv) if average is less than 35 then failed.

ex:

```
class Test
{
    public static void main(String[] args)
    {
        int m1=89,m2=37,m3=49,m4=53,m5=66,m6=72;

        int total=m1+m2+m3+m4+m5+m6;

        float avg=(float)total/6;

        System.out.println("Total :" +total);
        System.out.println("Average : " +avg);

        if(avg>=75)
            System.out.println("Grade : A grade");
        else if(avg>=50)
            System.out.println("Grade : B grade");
        else if(avg>=35)
            System.out.println("Grade : C grade");
        else
            System.out.println("Grade : Failed");
    }
}
```

3) Iteration statement

Iteration statement is used to execute the code repeatedly.

Iteration statement is possible by using loops.

We have four types of loops.

- i) do while loop
- ii) while loop
- iii) for loop
- iv) for each loop

- i) do while loop

It will execute the source code until our condition is true.

syntax:

```
do
{
```

```
- //code to be execute  
-  
}while(condition);
```

ex:

```
class Test  
{  
    public static void main(String[] args)  
    {  
        int i=1;  
        do  
        {  
            System.out.print(i+" "); // infinite 1  
        }  
        while(i<=10);  
    }  
}
```

ex:

```
class Test  
{  
    public static void main(String[] args)  
    {  
        int i=11;  
        do  
        {  
            System.out.print(i+" ");  
        }  
        while(i<=10);  
  
    }  
}
```

Note:

In do while loop our code will execute atleast for one time either our condition is true or false.

Q)Write a java program to display 10 natural numbers?

```
class Test  
{  
    public static void main(String[] args)  
    {  
        int i=1;  
        do  
        {  
            System.out.print(i+" ");  
            i++;  
        }  
    }  
}
```

```
        }
        while (i<=10);

    }
}
```

Q)Write a java program to display 10 natural numbers in descending order?

```
class Test
{
    public static void main(String[] args)
    {
        int i=10;
        do
        {
            System.out.print(i+" ");
            i--;
        }
        while (i>=1);

    }
}
```

Q)Write a java program to perform sum of 10 natural numbers?

```
class Test
{
    public static void main(String[] args)
    {
        int i=1,sum=0;
        do
        {
            sum=sum+i;
            i++;
        }
        while (i<=10);

        System.out.println("sum of 10 natural numbers is =" +sum);
    }
}
```

Q)Write a java program to find out factorial of a given number?

input: n=5

output: 120 (5*4*3*2*1)

```
import java.util.Scanner;
class Test
{
```

```

public static void main(String[] args)
{
    Scanner sc=new Scanner(System.in);

    System.out.println("Enter the number :");
    int n=sc.nextInt();

    int i=n,fact=1;
    do
    {
        fact=fact*i;
        i--;
    }
    while (i>=1);

    System.out.println("Factorial of a given number is =" +fact);
}
}

```

Q) Write a java program to display multiplication table of a given number?

ex:

```

import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the number :");
        int n=sc.nextInt();

        int i=1;
        do
        {
            System.out.println(n+" * "+i+" = "+n*i);
            i++;
        }
        while (i<=10);
    }
}

```

ii) while loop

It will execute the source code until our condition is true.

syntax: while(condition)

```
{
}
```

```
- //code to be execute  
-  
}
```

ex:

```
----  
class Test  
{  
    public static void main(String[] args)  
    {  
        int i=1;  
  
        while(i<=10)  
        {  
            System.out.print(i+" "); //infinite 1  
        }  
    }  
}
```

ex:

```
class Test  
{  
    public static void main(String[] args)  
    {  
        int i=11;  
        while(i<=10)  
        {  
            System.out.print(i+" "); //nothing  
        }  
    }  
}
```

Q)Write a java program to display 100 natural numbers?

```
class Test  
{  
    public static void main(String[] args)  
    {  
        int i=1;  
        while(i<=100)  
        {  
            System.out.print(i+" ");  
            i++;  
        }  
    }  
}
```

Q)Write a java program to perform sum of 10 natural numbers?

```
class Test
{
    public static void main(String[] args)
    {
        int i=1,sum=0;

        while(i<=10)
        {
            sum=sum+i;
            i++;
        }

        System.out.println(sum);
    }
}
```

Q)Write a java program to find out factorial of a given number ?

```
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the number :");
        int n=sc.nextInt();

        int i=n,fact=1;
        while(i>=1)
        {
            fact=fact*i;
            i--;
        }
        System.out.println(fact);
    }
}
```

Q)Write a java program to display multiplication table of a given number?

```
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
```

```

System.out.println("Enter the number :");
int n=sc.nextInt() // 5

int i=1;
while(i<=10)
{
    System.out.println(n+" * "+i+" = "+n*i);
    i++;
}
}

```

Q)Write a java program to perform sum of digits of a given number?

input: 123

output: 6 (1+2+3)

import java.util.Scanner;

class Test

{

 public static void main(String[] args)

 {

 Scanner sc=new Scanner(System.in);

 System.out.println("Enter the number :");

 int n=sc.nextInt() // 123

 int rem,sum=0;

 while(n>0)

 {

 rem=n%10;

 sum=sum+rem;

 n=n/10;

 }

 System.out.println("sum of digits of a given number is =" +sum);

 }

}

Q)Write a java program to find out given number is armstrong or not?

input: 153 (1*1*1+5*5*5+3*3*3) = (1+125+27) = (153)

output: It is a amrstrong number

import java.util.Scanner;

class Test

{

 public static void main(String[] args)

```

{
    Scanner sc=new Scanner(System.in);

    System.out.println("Enter the number :");
    int n=sc.nextInt() // 123

    int temp=n;

    int rem,sum=0;

    while(n>0)
    {
        rem=n%10;
        sum=sum+rem*rem*rem;
        n=n/10;
    }

    if(temp==sum)
        System.out.println("It is armstrong number");
    else
        System.out.println("It is not armstrong number");

    }
}

```

Q)Write a java program to find out reverse of a given number?

```

input: 123
output: 321
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the number :");
        int n=sc.nextInt() // 123

        int rem,rev=0;
        while(n>0)
        {
            rem=n%10;
            rev=rev*10+rem;
            n=n/10;
        }
    }
}

```

```

        System.out.println("Reverse of a given number is =" + rev);
    }
}

```

Q) Write a java program to check given number is palindrome or not?

input: 121

output: It is a palindrome number

import java.util.Scanner;

class Test

{

 public static void main(String[] args)

 {

 Scanner sc=new Scanner(System.in);

 System.out.println("Enter the number :");

 int n=sc.nextInt(); // 123

 int temp=n;

 int rem,rev=0;

 while(n>0)

 {

 rem=n%10;

 rev=rev*10+rem;

 n=n/10;

 }

 if(temp==rev)

 System.out.println("It is a palindrome number");

 else

 System.out.println("It is not a palindrome number");

 }

}

iii) for loop

It will execute the source code until our condition is true.

syntax:

for(initialization;condition;incrementation/decrementation)

{

-

- //code to be execute

-

}

ex:

class Test

```

{
    public static void main(String[] args)
    {
        for(int i=1;i<=10;i++)
        {
            System.out.print(i+" ");
        }
    }
}

```

ex:

```

class Test
{
    public static void main(String[] args)
    {
        for(int i=1;i<=10;i++)
        {
            System.out.print(i+" ");
            i--;
        }
    }
}

```

Q)Write a java program to display sum of even numbers from 1 to 10?

$$\text{sum} = (30) = 2 + 4 + 6 + 8 + 10$$

```

class Test
{
    public static void main(String[] args)
    {
        int sum=0;
        for(int i=1;i<=10;i++)
        {
            if(i%2==0)
            {
                sum+=i;
            }
        }
        System.out.println("sum of even numbers is =" + sum);
    }
}

```

Q)Write a java program to display number of even and odd numbers from 1 to 10?

```

class Test
{
    public static void main(String[] args)
    {
}
}

```

```

{
    int even=0,odd=0;
    for(int i=1;i<=10;i++)
    {
        if(i%2==0)
        {
            even++;
        }
        else
        {
            odd++;
        }
    }
    System.out.println("No of evens :" +even);
    System.out.println("No of odds :" +odd);
}
}

```

Q)Write a java program to display fibonacci series of a given number?

fibonacci series : 0 1 1 2 3 5 8

```

import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the number :");
        int n=sc.nextInt(); //5

        int a=0,b=1,c;

        System.out.print(a+" "+b+" ");

        for(int i=2;i<=n;i++)
        {
            c=a+b;
            System.out.print(c+" ");
            a=b;
            b=c;
        }
    }
}

```

Q)Write a java program to check given number is prime or not?

prime numbers:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.....

```
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the number :");
        int n=sc.nextInt(); //5

        boolean flag=true;

        for(int i=2;i<=n/2;i++)
        {
            if(n%i==0)
            {
                flag=false;
                break;
            }
        }
        if(flag==true)
            System.out.println("It is a prime number");
        else
            System.out.println("It is not a prime number");
    }
}
```

Q)Write a java program to check given number is perfect or not?

input: 6

output: It is perfect number

```
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the number :");
        int n=sc.nextInt(); //5

        int sum=0;
```

```

for(int i=1;i<n;i++)
{
    if(n%i==0)
    {
        sum=sum+i;
    }
}
if(n==sum)
    System.out.println("It is a perfect number");
else
    System.out.println("It is not a perfect number");
}
}

```

Q)Write a java program to find out GCD(Greatest Common Divisor) of two numbers?

input: 12 18

output: 6

class Test

```

{
    public static void main(String[] args)
    {
        int a=12,b=18,gcd=0;

        for(int i=1;i<12 || i<18;i++)
        {
            if(a%i==0 && b%i==0)
            {
                gcd=i;
            }
        }

        System.out.println("GCD of two numbers is =" +gcd);
    }
}

```

Q)Write a java program to display list of prime numbers from 1 to 100?

output: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.

class Test

```

{
    public static void main(String[] args)
    {
        for(int i=2;i<=100;i++)
        {
            boolean flag=true;

            for(int j=2;j<i;j++)

```

```

    {
        if(i%j==0)
        {
            flag=false;
            break;
        }
        if(flag==true)
            System.out.print(i+" ");
    }
}

```

Various ways to declare methods in java

There are four ways to declare methods in java.

- 1) No returntype With no argument method
- 2) No returntype With argument method
- 3) With returntype With no argument method
- 4) With returntype With argument method

- 1) No returntype With no argument method

If no arguments then we need to ask input values inside callie method.

Q)Write a java program to perform sum of two numbers using no returntype with no argument method?

```

import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        //caller method
        sum();
    }

    //callie method
    public static void sum()
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the first number :");
        int a=sc.nextInt();

        System.out.println("Enter the second number :");
        int b=sc.nextInt();

        int c=a+b;
    }
}

```

```
        System.out.println("sum of two numbers is =" + c);

    }

}
```

3) With returntype With no argument method

A returntype is completely depends upon output datatype.

Q) Write a java program to perform sum of two numbers using with returntype with no argument method?

```
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        //caller
        int k=sum();
        System.out.println("sum of two numbers is =" + k);
    }
    //callie method
    public static int sum()
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the First Number :");
        int a=sc.nextInt();
        System.out.println("Enter the Second Number :");
        int b=sc.nextInt();

        int c=a+b;

        return c;
    }
}
```

2) No returntype With argument method

Arguments are completely depends upon number of inputs.

Q) Write a java program to perform sum of two numbers using no returntype with argument method?

```
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
```

```

        System.out.println("Enter the first number :");
        int a=sc.nextInt();

        System.out.println("Enter the second number :");
        int b=sc.nextInt();

        //caller method
        sum(a,b);
    }
    public static void sum(int a,int b)
    {
        int c=a+b;
        System.out.println("sum of two numbers is =" +c);
    }
}

```

4) With returntype With argument method

Q) Write a java program to find out sum of two numbers using with returntype with argument method?

```

import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the first number :");
        int a=sc.nextInt();

        System.out.println("Enter the second number :");
        int b=sc.nextInt();

        //caller method
        System.out.println("sum of two numbers is =" +sum(a,b));
    }
    //callie method
    public static int sum(int a,int b)
    {
        int c=a+b;

        return c;
    }
}

```

Assignment

Q) Write a java program to find out factorial of a given number ?

Recursion In Java

A method which called itself for many number of times is called recursion.
Recursion is similar to loopings.

Whenever we use recursion, we should not use loops.

Q) Write a java program to perform sum of two numbers without using arithmetic operator ?

```
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the first number :");
        int a=sc.nextInt();

        System.out.println("Enter the second number :");
        int b=sc.nextInt();

        //caller method
        System.out.println("sum of two numbers is =" +sum(a,b));
    }

    //callie method
    public static int sum(int a,int b)
    {
        if(a==0)
            return b;

        return sum(--a,++b);
    }
}
```

Q) Write a java program to display 10 natural numbers without using loops?

```
class Test
{
    public static void main(String[] args)
    {
        //caller
        display(1);
    }

    //callie method
    public static void display(int i)
    {
        if(i<=10)
        {
            System.out.print(i+" ");
            display(i+1);
        }
    }
}
```

```

        }
    }
}

Q)Write a java program to display Nth element of fibonacci series ?
fibonacci series : 0 1 1 2 3 5 8
input: 4
output: 2
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the number : ");
        int n=sc.nextInt();//4

        //caller method
        System.out.println("Nth element of fibonacci series is =" +fib(n));
    }
    //callie method
    public static int fib(int n)
    {
        if(n==0 || n==1)
            return 0;
        if(n==2)
            return 1;

        return fib(n-1)+fib(n-2);
    }
}

```

Q)Write a java program to find out factorial of a given number using recursion?

input: 5
output: 120

```

import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the number : ");
        int n=sc.nextInt();//5

        //caller
        System.out.println(factorial(n));
    }
}

```

```

    }
    //callie method
    public static int factorial(int n)
    {
        if(n<0)
            return -1;
        if(n==0)
            return 1;
        return n*factorial(n-1);
    }
}

```

Q)Write a java program to check given number is palindrome or not using recursion?

```

class Test
{
    public static void main(String[] args)
    {
        int num=121;
        int original=num;
        int reversed=0;

        //caller method
        if(isPalindrome(num,original,reversed))
            System.out.println("It is a palindrome number");
        else
            System.out.println("It is not a palindrome number");
    }
    //callie method
    public static boolean isPalindrome(int num,int original,int reversed)
    {
        if(num==0)
            return original==reversed;

        reversed=reversed*10+num%10;

        return isPalindrome(num/10,original,reversed);
    }
}

```

LOOP Patterns

1)
1 1 1 1
2 2 2 2
3 3 3 3
4 4 4 4

```
class Test
{
    public static void main(String[] args)
    {
        //rows
        for(int i=1;i<=4;i++)
        {
            //cols
            for(int j=1;j<=4;j++)
            {
                System.out.print(i+" ");
            }
            //new line
            System.out.println("");
        }
    }
}
```

2)
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4

```
class Test
{
    public static void main(String[] args)
    {
        //rows
        for(int i=1;i<=4;i++)
        {
            //cols
            for(int j=1;j<=4;j++)
            {
                System.out.print(j+" ");
            }
            //new line
            System.out.println("");
        }
    }
}
```

```
* * * *
* * * *
* * * *
* * * *

class Test
{
    public static void main(String[] args)
    {
        //rows
        for(int i=1;i<=4;i++)
        {
            //cols
            for(int j=1;j<=4;j++)
            {
                System.out.print("* ");
            }
            //new line
            System.out.println("");
        }
    }
}

4)
4 4 4 4
3 3 3 3
2 2 2 2
1 1 1 1

class Test
{
    public static void main(String[] args)
    {
        //rows
        for(int i=4;i>=1;i--)
        {
            //cols
            for(int j=1;j<=4;j++)
            {
                System.out.print(i+" ");
            }
            //new line
            System.out.println("");
        }
    }
}
```

```

        }
    }

5)
A A A A
B B B B
C C C C
D D D D

class Test
{
    public static void main(String[] args)
    {
        //rows
        for(char i='A';i<='D';i++)
        {
            //cols
            for(char j='A';j<='D';j++)
            {
                System.out.print(i+" ");
            }
            //new line
            System.out.println("");
        }
    }
}

```

6)

```

D D D D
C C C C
B B B B
A A A A

```

ex:

```

class Test
{
    public static void main(String[] args)
    {
        //rows
        for(char i='D';i>='A';i--)
        {
            //cols
            for(char j='A';j<='D';j++)
            {
                System.out.print(i+" ");
            }
        }
    }
}

```

```

        }
        //new line
        System.out.println("");
    }
}

7)
* * * *
*   *
*   *
* * * *

class Test
{
    public static void main(String[] args)
    {
        //rows
        for(int i=1;i<=4;i++)
        {
            //cols
            for(int j=1;j<=4;j++)
            {
                if(i==1 || i==4 || j==1 || j==4)
                    System.out.print("* ");
                else
                    System.out.print(" ");
            }
            //new line
            System.out.println("");
        }
    }
}

8)
* - - -
- * - -
- - * -
- - - *

```

```

class Test
{
    public static void main(String[] args)
    {

```

```

//rows
for(int i=1;i<=4;i++)
{
    //cols
    for(int j=1;j<=4;j++)
    {
        if(i==j)
            System.out.print("* ");
        else
            System.out.print("- ");
    }
    //new line
    System.out.println("");
}
}

9)
* - - -
- * - *
- - * - -
- * - *
* - - -

```

```

class Test
{
    public static void main(String[] args)
    {
        //rows
        for(int i=1;i<=5;i++)
        {
            //cols
            for(int j=1;j<=5;j++)
            {
                if(i==j || i+j==6)
                    System.out.print("* ");
                else
                    System.out.print("- ");
            }
            //new line
            System.out.println("");
        }
    }
}

```

```

*
*
*
* * * * *
*
*

class Test
{
    public static void main(String[] args)
    {
        //rows
        for(int i=1;i<=5;i++)
        {
            //cols
            for(int j=1;j<=5;j++)
            {
                if(i==3 || j==3)
                    System.out.print("* ");
                else
                    System.out.print(" ");
            }
            //new line
            System.out.println("");
        }
    }
}

```

11)

```

1 1 1
1 0 1
1 1 1

```

```

class Test
{
    public static void main(String[] args)
    {
        //rows
        for(int i=1;i<=3;i++)
        {
            //cols
            for(int j=1;j<=3;j++)
            {
                if(i==2 && j==2)
                    System.out.print("0 ");
                else

```

```

        System.out.print("1 ");
    }
    //new line
    System.out.println("");
}
}
}
```

Left Side Loop patterns

1)
1
2 2
3 3 3
4 4 4 4

```

class Test
{
    public static void main(String[] args)
    {
        //rows
        for(int i=1;i<=4;i++)
        {
            //cols
            for(int j=1;j<=i;j++)
            {
                System.out.print(i+" ");
            }
            //new line
            System.out.println("");
        }
    }
}
```

2)
1
1 2
1 2 3
1 2 3 4

```

class Test
{
    public static void main(String[] args)
    {
        //rows
        for(int i=1;i<=4;i++)
```

```

{
    //cols
    for(int j=1;j<=i;j++)
    {
        System.out.print(j+" ");
    }
    //new line
    System.out.println("");
}

}

3)
*
*
* *
* * *
* * * *

class Test
{
    public static void main(String[] args)
    {
        //rows
        for(int i=1;i<=4;i++)
        {
            //cols
            for(int j=1;j<=i;j++)
            {
                System.out.print("* ");
            }
            //new line
            System.out.println("");
        }
    }
}

4)
4 4 4 4
3 3 3
2 2
1

class Test
{

```

```
public static void main(String[] args)
{
    //rows
    for(int i=4;i>=1;i--)
    {
        //cols
        for(int j=1;j<=i;j++)
        {
            System.out.print(i+" ");
        }
        //new line
        System.out.println("");
    }

}
}

5)
*
* *
* * *
* * * *
* *
* *
*
```

```
class Test
{
    public static void main(String[] args)
    {
        //rows
        for(int i=1;i<=4;i++)
        {
            //cols
            for(int j=1;j<=i;j++)
            {
                System.out.print("* ");
            }
            //new line
            System.out.println("");
        }
        //descending
        //rows
        for(int i=3;i>=1;i--)
        {
            //cols
```

```

        for(int j=1;j<=i;j++)
        {
            System.out.print("* ");
        }
        //new line
        System.out.println("");
    }

}

6)
1
2 3
4 5 6
7 8 9 0
class Test
{
    public static void main(String[] args)
    {
        int k=1;
        //rows
        for(int i=1;i<=4;i++)
        {
            //cols
            for(int j=1;j<=i;j++)
            {
                if(k<=9)
                    System.out.print(k++ + " ");
                else
                    System.out.println("0 ");
            }
            //new line
            System.out.println("");
        }
    }
}

```

Assignment

Q) Write a java program to display below loop pattern?

```

2
3 5
7 11 13
17 19 23 29

```

Right Side Elements

1)

```
    1
   2 2
  3 3 3
 4 4 4 4
```

```
class Test
{
    public static void main(String[] args)
    {
        //rows
        for(int i=1;i<=4;i++)
        {
            //space
            for(int j=4;j>i;j--)
            {
                System.out.print(" ");
            }

            //elements
            for(int j=1;j<=i;j++)
            {
                System.out.print(i+" ");
            }
            //new line
            System.out.println("");
        }
    }
}
```

2)

```
 4 4 4 4
  3 3 3
  2 2
  1
```

```
class Test
{
    public static void main(String[] args)
    {
        //rows
        for(int i=4;i>=1;i--)
        {
            //space
            for(int j=4;j>i;j--)
```

```

        {
            System.out.print(" ");
        }

        //elements
        for(int j=1;j<=i;j++)
        {
            System.out.print(i+" ");
        }
        //new line
        System.out.println("");
    }
}

```

3)

```

*
*
*
*
*
*
*
*
```

```

class Test
{
    public static void main(String[] args)
    {
        //ascending
        //rows
        for(int i=1;i<=4;i++)
        {
            //space
            for(int j=4;j>i;j--)
            {
                System.out.print(" ");
            }

            //elements
            for(int j=1;j<=i;j++)
            {
                System.out.print("* ");
            }
            //new line
            System.out.println("");
        }
    }
}
```

```

//descending
//rows
for(int i=3;i>=1;i--)
{
    //space
    for(int j=4;j>i;j--)
    {
        System.out.print(" ");
    }

    //elements
    for(int j=1;j<=i;j++)
    {
        System.out.print("* ");
    }
    //new line
    System.out.println("");
}
}

```

Pyramids

=====

1)

```

      1
     1 2 1
    1 2 3 2 1
   1 2 3 4 3 2 1

```

class Test

{

```

    public static void main(String[] args)
    {
        //rows
        for(int i=1;i<=4;i++)
        {
            //space
            for(int j=4;j>i;j--)
            {
                System.out.print(" ");
            }
            //left side element
            for(int j=1;j<=i;j++)
            {
                System.out.print(j+" ");
            }
        }
    }
}
```

```

        }
        //right side elements
        for(int j=i-1;j>=1;j--)
        {
            System.out.print(j+" ");
        }

        //new line
        System.out.println("");
    }
}

2)
1 2 3 4 3 2 1
1 2 3 2 1
1 2 1
1
class Test
{
    public static void main(String[] args)
    {
        //rows
        for(int i=4;i>=1;i--)
        {
            //space
            for(int j=4;j>i;j--)
            {
                System.out.print(" ");
            }
            //left side element
            for(int j=1;j<=i;j++)
            {
                System.out.print(j+" ");
            }
            //right side elements
            for(int j=i-1;j>=1;j--)
            {
                System.out.print(j+" ");
            }

            //new line
            System.out.println("");
        }
    }
}

```

```

3)
    *
    * *
    * * *
    * * * *
    * * * * *
    * * *
    *
ex:
class Test
{
    public static void main(String[] args)
    {
        //rows
        for(int i=1;i<=4;i++)
        {
            //space
            for(int j=4;j>i;j--)
            {
                System.out.print(" ");
            }
            //left side element
            for(int j=1;j<=i;j++)
            {
                System.out.print("* ");
            }
            //right side elements
            for(int j=i-1;j>=1;j--)
            {
                System.out.print("* ");
            }

            //new line
            System.out.println("");
        }

        //descending
        //rows
        for(int i=3;i>=1;i--)
        {
            //space
            for(int j=4;j>i;j--)
            {
                System.out.print(" ");
            }

```

```

        //left side element
        for(int j=1;j<=i;j++)
        {
            System.out.print("* ");
        }
        //right side elements
        for(int j=i-1;j>=1;j--)
        {
            System.out.print("* ");
        }

        //new line
        System.out.println("");
    }
}
}

```

Interview Question

Q) Write a java program to display below loop pattern?

```

1      1
1 2    2 1
1 2 3  3 2 1
1 2 3 4 4 3 2 1

```

class Test

```

{
    public static void main(String[] args)
    {
        int rows=4;

        //rows
        for(int i=1;i<=rows;i++)
        {
            //left side elements
            for(int j=1;j<=i;j++)
            {
                System.out.print(j+" ");
            }
            //space
            for(int j=1;j<=(rows-i)*2;j++)
            {
                System.out.print(" ");
            }
            //right side elements
            for(int j=i;j>=1;j--)
            {
                System.out.print(j+" ");
            }
        }
    }
}
```

```

        }
        //new line
        System.out.println("");
    }

}

```

Q)Write a java program to display below loop pattern?

```

1
2 1
1 2 3
4 3 2 1
class Test
{
    public static void main(String[] args)
    {
        int rows=4;

        for(int i=1;i<=rows;i++)
        {
            if(i%2==0)
            {
                for(int j=i;j>=1;j--)
                {
                    System.out.print(j+" ");
                }
            }
            else
            {
                for(int j=1;j<=i;j++)
                {
                    System.out.print(j+" ");
                }
            }
            //new line
            System.out.println("");
        }
    }
}

```

4)Jump Statement

Jump statement is used to jump from one section of code to another section.
We have two jump statements in java.

- i) break statement
- ii) continue statement

i) break statement

It is used to break the execution of loops and switch case.

For conditional statement we can use if condition.

```
syntax:      break;  
class Test  
{  
    public static void main(String[] args)  
    {  
        System.out.println("stmt1");  
        break;  
        System.out.println("stmt2");  
    }  
}
```

o/p:

break outside switch or loop

ex:2

```
class Test  
{  
    public static void main(String[] args)  
    {  
        System.out.println("stmt1");  
        if(true)  
        {  
            break;  
        }  
        System.out.println("stmt2");  
    }  
}
```

o/p:

break outside switch or loop

ex:3

```
class Test  
{  
    public static void main(String[] args)  
    {  
        for(int i=1;i<=10;i++)  
        {  
            if(i==5)
```

```

    {
        break;
    }
    System.out.print(i+" ");
}
}

```

ii) continue statement

It is used to continue the execution of loops.

For conditional statements we can use if condition.

syntax: continue;

ex:1

```

class Test
{
    public static void main(String[] args)
    {
        System.out.println("stmt1");
        continue;
        System.out.println("stmt2");
    }
}

```

o/p: continue outside of loop

ex:2

```

class Test
{
    public static void main(String[] args)
    {
        System.out.println("stmt1");
        if(true)
        {
            continue;
        }
        System.out.println("stmt2");
    }
}

```

o/p: continue outside of loop

ex:3

```

class Test
{

```

```

public static void main(String[] args)
{
    for(int i=1;i<=10;i++)
    {
        if(i==5)
        {
            continue;
        }
        System.out.print(i+" ");//1 2 3 4 6 7 8 9 10
    }
}

```

Interview Program

Q) Write a java program to display reverse of a given number in words?

input: 123

output: ThreeTwoOne

```

import java.util.Scanner;
class Test
{

```

```

    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the number :");
        int n=sc.nextInt();

        while(n>0)
        {
            switch(n%10)
            {
                case 0 : System.out.print("Zero");break;
                case 1 : System.out.print("One");break;
                case 2 : System.out.print("Two");break;
                case 3 : System.out.print("Three");break;
                case 4 : System.out.print("Four");break;
                case 5 : System.out.print("Five");break;
                case 6 : System.out.print("Six");break;
                case 7 : System.out.print("Seven");break;
                case 8 : System.out.print("Eight");break;
                case 9 : System.out.print("Nine");break;
            }
            n=n/10;
        }
    }
}
```

Assignment

Write a java program to display sum of digits of a given number and display in reverse order?

input: 987

output: 42

Arrays

A normal variable we can store only one value at a time.

In order to store more than one value in a single variable then we need to use arrays.

Array is a collection of homogeneous data elements.

The main advantages of arrays are

1) We can represent multiple elements using single variable name.

ex: int[] arr={10,20,30};

2) Performance point of view arrays are recommended to use.

The main disadvantages of arrays are.

1) Arrays are fixed in size once if we create an array there is no chance of increasing or decreasing the size of an array.

2) To use array concept in advanced we should know what is the size of an array which is always not possible.

In java, arrays are divided into three types.

1) Single dimensional array

2) Double dimensional array / two dimensional array

3) Multi dimensional array / three dimensional array

Array declaration

At the time of array declaration we should not specify array size.

Arrays

-----	-----	-----
Single dimensional array	Double dimensional array	Multidimensional array

int[] arr;

int []arr;

int arr[];

int[][] arr;

int [][]arr;

int arr[][];

int[] []arr;

int[] arr[];

int []arr[];

int[][][] arr;

int [][] []arr;

int arr[][][];

int[] [] []arr;

int[] arr[][];

int[] []arr[];

int [] [] []arr[];

int [] []arr [];

Array Creation

In java, every array consider as an object.Hence we will use new operator to create an array.

ex: int[] arr=new int[3];

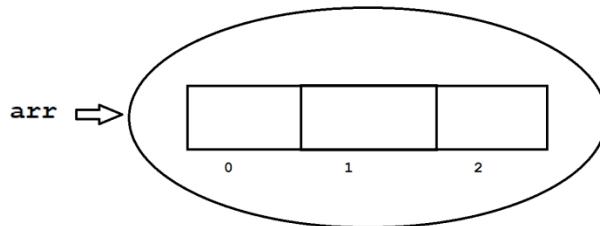


Diagram : java20.1

Rules to construct an array

Rule1:

At the time of array creation compulsory we need to specify array size.

ex: int[] arr=new int[3]; //valid

int[] arr=new int[]; //C.T.E array dimension missing

Rule2:

It is legal to have an array size with zero.

ex: int[] arr=new int[0];

System.out.println(arr.length); //0

Rule3:

We can't give negative number as an array size otherwise we will get

NegativeArraySizeException.

ex: int[] arr=new int[-3]; // R.E NegativeArraySizeException

Rule4:

The allowed datatype for an array size is byte,short,int and char .If we take other datatypes then we will get compile time error.

ex: byte b=10;

int[] arr=new int[b];

int[] arr=new int['a'];

int[] arr=new int[10.5f]; //invalid

Rule5:

The maximum length we can take for an array is maximum length of int.

ex: int[] arr=new int[2147483647];

Array Initialization

Once if we create an array then every array element will be initialized with default values.

It is possible to change default values with customized values also.

ex:

int[] arr=new int[3];

arr[0]=10;

arr[1]=20;

arr[2]=30;

arr[3]=40; // R.E ArrayIndexOutOfBoundsException

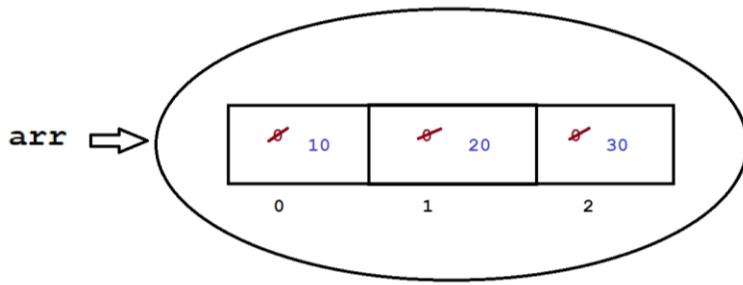


Diagram: java20.2

Array Declaration , Creation and Initialization using single line

```
int[] arr;
arr = new int[3];
arr[0]=10;
arr[1]=20;
arr[2]=30;      =====>    int[] arr={10,20,30};
                  =====>    char[] carr={'a','b','c'};
                  =====>    String[] sarr={"hi","hello","bye"};
```

Q)What is the difference between length and length() ?

length

It is a final variable which is applicable for arrays.

It will return size of an array.

```
ex:    class Test
{
    public static void main(String[] args)
    {
        int[] arr=new int[3];
        System.out.println(arr.length());//3
    }
}
```

length()

It is a predefined method applicable for String objects.

It will return number of characters present in String.

```
ex:    class Test
{
    public static void main(String[] args)
    {
        String str="bhaskar";
        System.out.println(str.length());//7
    }
}
```

Q)Write a java program to accept array elements and display them ?

```
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the array size :");
        int size=sc.nextInt(); //4

        int[] arr=new int[size];

        //inserting elements
        for(int i=0;i<arr.length;i++)
        {
            System.out.println("Enter the element :");
            arr[i]=sc.nextInt();
        }

        //display elements
        for(int i=0;i<arr.length;i++)
        {
            System.out.print(arr[i]+ " ");
        }

    }
}
```

Q)Write a java program to display array elements ?

input: arr = 4 8 1 3 9

```
class Test
{
    public static void main(String[] args)
    {
        int[] arr={4,8,1,3,9};

        //normal for loop
        for(int i=0;i<arr.length;i++)
        {
            System.out.print(arr[i]+ " ");
        }
        System.out.println("\n=====");

        //for each loop
        for(int ele:arr)
```

```

        {
            System.out.print(ele+" ");
        }
    }
}

```

Q)Write a java program to perform sum array elements ?

input: arr = 4 8 1 3 9

output: 25

```

class Test
{
    public static void main(String[] args)
    {

        int[] arr={4,8,1,3,9};

        //sum of array elements
        int sum=0;
        for(int ele:arr)
        {
            sum+=ele;
        }

        System.out.println(sum);
    }
}

```

Q)Write a java program to display array elements in reverse order?

input: arr = 4 8 1 3 9

output: 9 3 1 8 4

```

class Test
{
    public static void main(String[] args)
    {

        int[] arr={4,8,1,3,9};

        for(int i=arr.length-1;i>=0;i--)
        {
            System.out.print(arr[i]+" ");
        }
    }
}

```

Q)Write a java program to display array elements in ascending/sorting order?

input: arr = 4 8 1 3 9

output: 1 3 4 8 9

approach1:

```
-----  
import java.util.Arrays;  
class Test  
{  
    public static void main(String[] args)  
    {  
        int[] arr={4,8,1,3,9};  
  
        Arrays.sort(arr); // 1 3 4 8 9  
  
        //display elements  
        for(int ele:arr)  
        {  
            System.out.print(ele+" ");  
        }  
    }  
}
```

Approach2

```
-----  
class Test  
{  
    public static void main(String[] args)  
    {  
        int[] arr={4,8,1,3,9};  
  
        //ascending logic  
        for(int i=0;i<arr.length;i++)  
        {  
            for(int j=0;j<arr.length;j++)  
            {  
                if(arr[i]<arr[j])  
                {  
                    int temp=arr[i];  
                    arr[i]=arr[j];  
                    arr[j]=temp;  
                }  
            }  
        }  
  
        //display elements  
        for(int ele:arr)  
        {
```

```

        System.out.print(ele+" ");
    }
}
}
```

Q)Write a java program to display array elements in descending order?

input: arr = 4 8 1 3 9

output: 9 8 4 3 1

approach1:

```

import java.util.Arrays;
class Test
{
    public static void main(String[] args)
    {
```

```
        int[] arr={4,8,1,3,9};
```

```
        Arrays.sort(arr);//1 3 4 8 9
```

```
        //display elements
```

```
        for(int i=arr.length-1;i>=0;i--)
        {
            System.out.print(arr[i]+" ");
        }
```

```
}
```

approach2

```
class Test
```

```
{
```

```
    public static void main(String[] args)
```

```
{
```

```
        int[] arr={4,8,1,3,9};
```

```
        //descending logic
```

```
        for(int i=0;i<arr.length;i++)
```

```
{
```

```
            for(int j=0;j<arr.length;j++)
```

```
{
```

```
                if(arr[i]>arr[j])
```

```
{
```

```
                    int temp=arr[i];
```

```

        arr[i]=arr[j];
        arr[j]=temp;
    }
}

//display elements
for(int ele:arr)
{
    System.out.print(ele+" ");
}
}

```

Q)Write a java program to display least element from given array?

input: arr = 4 8 1 3 9

output: 1

approach1

```

import java.util.Arrays;
class Test
{
    public static void main(String[] args)
    {
        int[] arr={4,8,1,3,9};

        Arrays.sort(arr);//1 3 4 8 9

        System.out.println(arr[0]);
    }
}

```

Approach2

```

class Test
{
    public static void main(String[] args)
    {

        int[] arr={4,8,1,3,9};

        int small=arr[0];

        for(int i=0;i<arr.length;i++)
        {
            if(arr[i]<small)

```

```

        {
            small=arr[i];
        }
    }
    System.out.println(small);
}
}

```

Q)Write a java program to display highest element from given array?

input: arr = 4 8 1 3 9

output: 9

approach1

```

import java.util.Arrays;
class Test
{
    public static void main(String[] args)
    {
        int[] arr={4,8,1,3,9};

        Arrays.sort(arr);

        System.out.println(arr[arr.length-1]);//9
    }
}

```

approach2

```

class Test
{
    public static void main(String[] args)
    {

        int[] arr={4,8,1,3,9};

        int big=arr[0];

        for(int i=0;i<arr.length;i++)
        {
            if(arr[i]>big)
            {
                big=arr[i];
            }
        }
        System.out.println(big);
    }
}

```

Q)Write a java program to display duplicate elements from given array?

input: 2 4 5 7 2 3 9 3 1 1

output: 2 3 1

class Test

{

```
    public static void main(String[] args)
    {
        int[] arr={2,4,5,7,2,3,9,3,1,1};

        //duplicate elements
        for(int i=0;i<arr.length;i++)
        {
            for(int j=i+1;j<arr.length;j++)
            {
                if(arr[i]==arr[j])
                    System.out.print(arr[i]+" ");
            }
        }
    }
}
```

Q)Write a java program to find out unique elements from given array?

input: 2 3 5 6 7 2 9 3 1 5

output: 6 7 9 1

class Test

{

```
    public static void main(String[] args)
    {
        int[] arr={2,3,5,6,7,2,9,3,1,5};
        //unique elements
        for(int i=0;i<arr.length;i++)
        {
            int cnt=0;
            for(int j=0;j<arr.length;j++)
            {
                if(arr[i]==arr[j])
                {
                    cnt++;
                }
            }
            if(cnt==1)
                System.out.print(arr[i]+" ");
        }
    }
}
```

Q)Write a java program to find out most repeating element from given array?

input: 1 3 5 2 1 9 1 1 3 3 7 1 6

output: 1 is repeating for 5 times

class Test

{

```
    public static void main(String[] args)
    {
        int[] arr={1,3,5,2,1,9,1,1,3,3,7,1,6};

        int maxCount=0;
        int element=0;

        for(int i=0;i<arr.length;i++)
        {
            int cnt=0;
            for(int j=0;j<arr.length;j++)
            {
                if(arr[i]==arr[j])
                {
                    cnt++;
                }
            }
            if(maxCount<cnt)
            {
                maxCount=cnt;
                element=arr[i];
            }
        }
        System.out.println(element+" is repeating for "+maxCount+" times");
    }
}
```

Q)Write a java program to display prime elements from given arrays?

input: 5 9 2 6 21 25 29

output: 5 2 29

class Test

{

```
    public static void main(String[] args)
    {
        int[] arr={5,9,2,6,21,25,29};

        for(int ele:arr)
        {
            boolean flag=true;

            for(int j=2;j<=ele/2;j++)

```

```

        {
            if(ele%j==0)
            {
                flag=false;
                break;
            }
        }
        if(flag==true)
            System.out.print(ele+" ");
    }
}

```

Q)Write a java program to segregate 0's and 1's?

input: 1 0 1 1 0 0 1 0 0 1

output: 0 0 0 0 1 1 1 1 1

approach1

```

import java.util.Arrays;
class Test
{
    public static void main(String[] args)
    {
        int[] arr={1,0,1,1,0,0,1,0,0,1};

        Arrays.sort(arr);

        for(int ele:arr)
        {
            System.out.print(ele+" ");
        }
    }
}

```

approach2

```

class Test
{
    public static void main(String[] args)
    {
        int[] arr={1,0,1,1,0,0,1,0,0,1};

        int[] resArr=new int[arr.length];

        //inserting 0's
        int j=0;

```

```

for(int ele:arr)
{
    if(ele==0)
    {
        resArr[j++]=0;
    }
}

//inserting 1's
while(j<arr.length)
{
    resArr[j++]=1;
}

//display elements
for(int ele:resArr)
{
    System.out.print(ele+" ");
}
}
}

```

Q)Write a java program to find out missing element from given array?

input: 5 1 3 2 6 7

output: 4

class Test

```

{
    public static void main(String[] args)
    {
        int[] arr={5,1,3,2,6,7};

        int sum_of_arr_ele=arr.length+1;

        int sum=(sum_of_arr_ele*(sum_of_arr_ele+1))/2;

        for(int ele:arr)
        {
            sum=sum-ele;
        }

        System.out.println("Missing element is =" +sum);
    }
}

```

Q)Write a java program to find out leader element from given array?

input: 2 10 6 34 19 1 7

output: 7 19 34

class Test

```
{  
    public static void main(String[] args)  
    {  
        int[] arr={2,10,6,34,19,1,7};  
  
        int max=arr[arr.length-1];  
  
        System.out.print(max+" ");  
  
        for(int i=arr.length-2;i>=0;i--)  
        {  
            if(arr[i]>max)  
            {  
                max=arr[i];  
                System.out.print(max+" ");  
            }  
        }  
    }  
}
```

Q)Write a java program to perform sum of array elements?

input: arr1 = 2 4 7 1 3

arr2 = 9 8 6 4 3

output: 11 12 13 5 6

class Test

```
{  
    public static void main(String[] args)  
    {  
        int[] arr1 ={2,4,7,1,3};  
        int[] arr2 ={9,8,6,4,3};  
        int[] resArr=new int[arr1.length];  
        for(int i=0;i<arr1.length;i++)  
        {  
            resArr[i]=arr1[i]+arr2[i];  
        }  
        //for each loop  
        for(int ele:resArr)  
        {  
            System.out.print(ele+" ");  
        }  
    }  
}
```

Q)Write a java program to delete first occurrence of a given element?

```
input: arr = 2 4 5 6 4 9 1 3 4
      ele = 4
output: 2 5 6 4 9 1 3 4
class Test
{
    public static void main(String[] args)
    {
        int[] arr={2,4,5,6,4,9,1,3,4};
        int ele=4;

        int[] resArr=new int[arr.length-1];

        int cnt=0;
        int j=0;
        for(int i=0;i<arr.length;i++)
        {
            if(arr[i]==ele && cnt==0)
            {
                cnt++;
                continue;
            }
            resArr[j++]=arr[i];
        }

        //display
        for(int i:resArr)
        {
            System.out.print(i+" ");
        }
    }
}
```

Q)write a java program to concatinate two arrays and display them in sorting order?

```
input: arr1 = 9 6 8 10 7
      arr2 = 1 5 4 2 3
output: 1 2 3 4 5 6 7 8 9 10
import java.util.Arrays;
class Test
{
    public static void main(String[] args)
    {
        int[] arr1 ={9,6,8,10,7};
        int[] arr2 ={1,5,4,2,3};

        int size1=arr1.length;
```

```

int size2=arr2.length;

arr1=Arrays.copyOf(arr1,size1+size2);

int j=0;
for(int i=size1;i<arr1.length;i++)
{
    arr1[i]=arr2[j++];
}

Arrays.sort(arr1);

//display the elements
for(int ele:arr1)
{
    System.out.print(ele+" ");
}
}
}

```

Q)Write a java program to insert given element on given index?

input: arr = 5 8 1 4 6 9

ele = 100

index = 3

output: 5 8 1 100 4 6 9

import java.util.Arrays;

class Test

{

public static void main(String[] args)

{

int[] arr ={5,8,1,4,6,9};

int ele = 100;

int index = 3;

arr=Arrays.copyOf(arr,arr.length+1);

for(int i=arr.length-1;i>=index;i--)

{

arr[i]=arr[i-1];

}

arr[index]=ele;

//display

for(int i:arr)

{

```

        System.out.print(i+" ");
    }
}
}

```

Two Dimensional Array

Two dimensional array is a combination of rows and columns.

Two dimensional array is implemented based on array of arrays approach but not in matrix form.

The main objective of two dimensional array is memory utilization.

Two dimensional array is used to develop business oriented applications, gaming applications and matrix type of applications.

We can declare two dimensional array as follow.

```

syntax:      datatype[][] variable_name= new int[rows][cols];
ex:       int[][] arr=new [3][3];
           Here we can store 9 elements.

```

Q)Write a java program to display array elements in matrix form?

```

import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the rows :");
        int rows=sc.nextInt(); //3

        System.out.println("Enter the cols :");
        int cols=sc.nextInt(); //3

        int[][] arr=new int[rows][cols];

        //insert elements
        for(int i=0;i<rows;i++)
        {
            for(int j=0;j<cols;j++)
            {
                System.out.println("Enter the element :");
                arr[i][j]=sc.nextInt();
            }
        }

        //display elements
        for(int i=0;i<rows;i++)
        {
            for(int j=0;j<cols;j++)

```

```

        {
            System.out.print(arr[i][j]+" ");
        }
        //new line
        System.out.println("");
    }

}

```

Q)Write a java program to perform sum of diagonal elements?

```

class Test
{
    public static void main(String[] args)
    {
        int[][] arr={{1,2,3}, {4,5,6}, {7,8,9}};
        //display elements
        int sum=0;
        for(int i=0;i<3;i++)
        {
            for(int j=0;j<3;j++)
            {
                if(i==j)
                {
                    sum=sum+arr[i][j];
                }
            }
        }
        System.out.println("sum of diagonal elements is =" +sum);
    }
}

```

Q)Write a java program to display sum of upper triangle elements?

```

class Test
{
    public static void main(String[] args)
    {
        int[][] arr={{1,2,3}, {4,5,6}, {7,8,9}};
        //display elements
        int sum=0;
        for(int i=0;i<3;i++)
        {
            for(int j=0;j<3;j++)
            {
                if(i<j)

```

```

        {
            sum=sum+arr[i][j];
        }
    }
System.out.println("sum of upper triangle elements is =" +sum);

}
}

```

Q)Write a java program to display sum of lower triangle elements?

```

class Test
{
    public static void main(String[] args)
    {
        int[][] arr={{1,2,3}, {4,5,6}, {7,8,9}};
        //display elements
        int sum=0;
        for(int i=0;i<3;i++)
        {
            for(int j=0;j<3;j++)
            {
                if(i>j)
                {
                    sum=sum+arr[i][j];
                }
            }
        }
        System.out.println("sum of lower triangle elements is =" +sum);
    }
}

```

Q)Write a java program to display array elements in spiral form ?

```

input: 1 2 3
      4 5 6
      7 8 9
output: 1 2 3 6 9 8 7 4 5
public class Test
{
    public static void main(String[] args)
    {
        int[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
        int rows = matrix.length;
        int cols = matrix[0].length;
    }
}

```

```

int top = 0;
int bottom = rows - 1;
int left = 0;
int right = cols - 1;

while (true)
{
    if (left > right)
    {
        break;
    }

    // Print top row
    for (int i = left; i <= right; i++) {
        System.out.print(matrix[top][i] + " ");
    }
    top++;

    if (top > bottom) {
        break;
    }

    // Print right column
    for (int i = top; i <= bottom; i++) {
        System.out.print(matrix[i][right] + " ");
    }
    right--;

    if (left > right) {
        break;
    }

    // Print bottom row
    for (int i = right; i >= left; i--)
    {
        System.out.print(matrix[bottom][i] + " ");
    }
    bottom--;

    if (top > bottom) {
        break;
    }

    // Print left column
    for (int i = bottom; i >= top; i--)
    {

```

```

        System.out.print(matrix[i][left] + " ");
    }
    left++;
} //while loop
}
}

```

Anonymous Array

Sometimes we will declare an array without name such type of nameless arrays are called anonymous array.

The main objective of anonymous array is just for instance use.

We can declare anonymous array as follow.

ex: new int[]{10,20,30};
new int[][]{{10,20,30},{40,50,60}};

```

public class Test
{
    public static void main(String[] args)
    {
        //caller method
        sum(new int[]{10,20,30});

    }
    //callie method
    public static void sum(int[] arr)
    {
        int sum=0;
        for(int i:arr)
        {
            sum+=i;
        }
        System.out.println(sum);
    }
}

```

ex:2

```

public class Test
{
    public static void main(String[] args)
    {
        //caller method
        System.out.println(sum(new int[]{10,20,30}));

    }
    //callie method
    public static int sum(int[] arr)

```

```

{
    int sum=0;
    for(int i:arr)
    {
        sum+=i;
    }
    return sum;
}
}

```

OOPS - OOPS stands for Object Oriented Programming System/Structure.

Object oriented technology

A technology which provides very good environment to represent our data in the form of objects is called object oriented technology.

A technology is said to be object oriented if it support following features.

ex: class, object, Abstraction, Encapsulation, Inheritance & Polymorphism

class

A class is a collection of data members and behaviours in a single unit.

Here data members means variables, fields and properties.

Here behaviours means methods, actions and characteristics.

In general , a class is a collection of variables and methods.

It is a blue print of an object.

We can declare a class as follow.

syntax:

```

optional
|
modifier class class_name <extends> Parent_class <implements> Interface_name
{
    -
    - // variables & methods
    -
}

```

A class will accept following modifiers.

ex: default, public, final & abstract

Q)What is the difference between default class and public class?

default class

If we declare any class as default then we can access that class within the package.

ex: class A

```

{
    -
    -
}

```

public class

If we declare any class as public then we can access that class within the package and outside of the package.

```
ex:    public class A
      {
      -
      -
      }
```

Q)What is **final class**?

If we declare any class as final creating child class is not possible.

or

If we declare any class as final then extending someother class is not possible.

```
ex:    final class A
      {
      }
      class B extends A // invalid
      {
      }
```

Q)What is **abstract class**?

If we declare any class as abstract then creating object for that class is not possible.

```
ex:    abstract class A
      {
      -
      -
      }
      A a=new A(); //invalid
```

object

It is a instance of a class.

Allocating memory for our data members is called instance.

Object is a outcome of a blue print.

Memory space will be created when we create an object.

We can declare object as follow.

syntax: class_name reference_variable=new constructor();

```
ex:    Test t = new Test();
```

It is possible to declare more then one object in a single class.

```
class Test
{
    public static void main(String[] args)
    {
        Test t1=new Test();
        Test t2=new Test();
        Test t3=new Test();
```

```

        System.out.println(t1.hashCode());
        System.out.println(t2.hashCode());
        System.out.println(t3.hashCode());

        System.out.println(t1); //Test@Hexavalue
        System.out.println(t2.toString());
        System.out.println(t3.toString());
    }
}

```

hashCode()

A hashCode() method present in Object class.

For every object , JVM will create a unique identification number i.e hash code.

In order to read hash code of an object we need to use hashCode() method.

```
Test t=new Test();
```

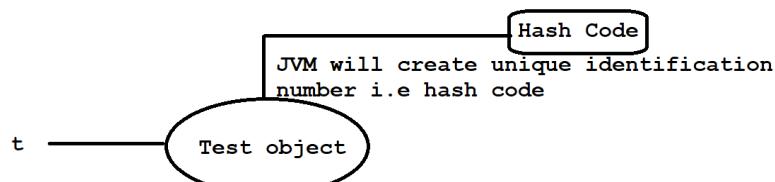


Diagram: java23.1

toString()

A toString() method present in Object class.

Whenever we are trying to display any object reference directly or indirectly toString() method will be executed.

Object class

Object is a class which is present in java.lang package.

It is a parent class for every java class.

Object class contains following methods.

ex: cmd> javap java.lang.Object

- hashCode()
- notify()
- notifyAll()
- getClass()
- toString()
- equals()
- and etc.

Data Hiding

Our internal data should not go out directly.

It means outside person must not access our data directly.

Using private modifier we can achieve data hiding concept.

The main objective of data hiding is to provide security.

```

ex: class Account
{
    private double balance;
}

```

Abstraction

Hiding internal implementation and highlighting the set of services is called abstraction.

Using abstract classes and interfaces we can implement Abstraction.

Best example of abstraction is GUI(Graphical User Interface) ATM machine where bank people will hide internal implementation and highlights the set of services like banking ,withdrawl, mini statement and etc.

The main advantages of abstraction are.

- 1)It gives security because it will hide internal implementation from the outsider.
- 2) Enhancement becomes more easy because without effecting enduser they can perform any changes in our internal system.
- 3) It provides flexibility to the enduser to use the system.
- 4) It improve maintainability of an application

Encapsulation

The process of encapsulating or grouping variables and it's associate methods in a single entity is called encapsulation.

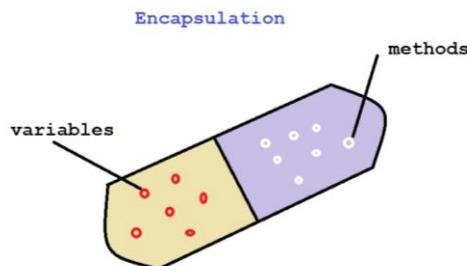


Diagram: java23.2

A class is said to be encapsulated class if it supports data hiding and abstraction.

In encapsulation,for every variable we need to write setter and getter method.

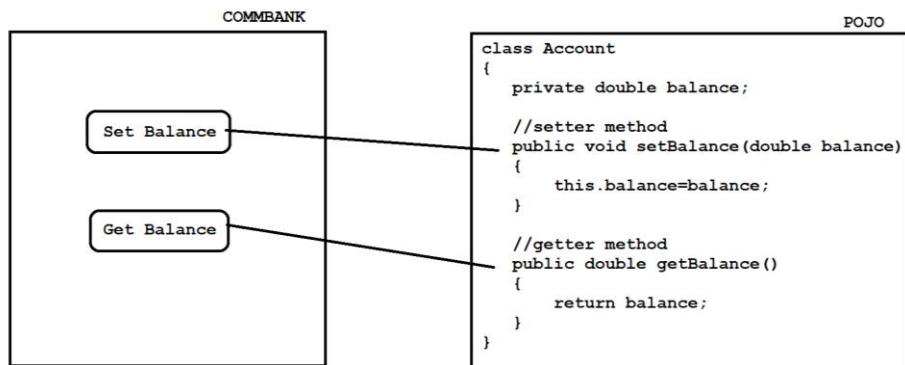


Diagram: java23.3

The main advantages of encapsulation are

- 1)It gives security.
- 2)Enhancement becomes more easy.
- 3)It provides flexibility to the enduser.
- 4)It improves maintainability of an application.

The main disadvantage of encapsulation is , it will increase the length of our code and slowdown the execution process.

Approach1

```
class Student
{
    private int studId;
    private String studName;
    private double studFee;

    //setter methods
    public void setStudId(int studId)
    {
        this.studId=studId;
    }
    public void setStudName(String studName)
    {
        this.studName=studName;
    }
    public void setStudFee(double studFee)
    {
        this.studFee=studFee;
    }

    //getter methods
    public int getStudId()
    {
        return studId;
    }
    public String getStudName()
    {
        return studName;
    }
    public double getStudFee()
    {
        return studFee;
    }

    public static void main(String[] args)
    {
        Student s=new Student();
```

```

        s.setStudId(101);
        s.setStudName("Alan");
        s.setStudFee(1000.0d);
        System.out.println("Student Id :" + s.getStudId());
        System.out.println("Student Name :" + s.getStudName());
        System.out.println("Student Fee :" + s.getStudFee());

    }
}

```

Approach2

```

class Student
{
    private int studId;
    private String studName;
    private double studFee;

    //setter methods
    public void setStudId(int studId)
    {
        this.studId=studId;
    }
    public void setStudName(String studName)
    {
        this.studName=studName;
    }
    public void setStudFee(double studFee)
    {
        this.studFee=studFee;
    }

    //getter methods
    public int getStudId()
    {
        return studId;
    }
    public String getStudName()
    {
        return studName;
    }
    public double getStudFee()
    {
        return studFee;
    }
}
class Test

```

```

{
    public static void main(String[] args)
    {
        Student s=new Student();
        s.setStudId(101);
        s.setStudName("Alan");
        s.setStudFee(1000.0d);
        System.out.println("Student Id :" +s.getStudId());
        System.out.println("Student Name :" +s.getStudName());
        System.out.println("Student Fee :" +s.getStudFee());
    }
}

```

*Note

Abstraction is used to hide the data.

Encapsulation is used to protect the data using access modifiers.

Interview Questions

Q) What is tightly encapsulated class?

A class is said to be tightly encapsulated class. If all the variables of that class must be private and we don't need to check these variables have setter or getter methods or not.

ex: class A

```

{
    private int i=10;
}
```

ex: class A

```

{
    private int i=10;
}
class B extends A
{
    private int j=20;
}
```

Q) What is the difference between POJO class and Java Bean class?

POJO class

POJO stands for Plain Old Java Object.

A class is said to be pojo class if it supports following two properties.

- 1) All variables must be private.
- 2) All the variables must have setter and getter methods.

JavaBean class

A class is said to be java bean class if it supports following four properties.

- 1) A class should be public.
- 2) A class should have zero-argument constructor.
- 3) All variables must be private.

4) All variables must have setter and getter methods.

*Note: Every java bean class is a pojo class but every pojo class is not a java bean class.

Is-A relationship

Is-A relationship is also known as inheritance.

By using extends keyword we can implements Is-A relationship.

The main objective of Is-A relationship is to provide reusability.

class Parent

{

```
    public void m1()
    {
        System.out.println("Parent-M1 Method");
    }
```

}

class Child extends Parent

{

```
    public void m2()
    {
        System.out.println("Child-M2 Method");
    }
```

}

class Test

{

```
    public static void main(String[] args)
    {
```

```
        Parent p=new Parent();
        p.m1();
        Child c=new Child();
        c.m1();
        c.m2();
        Parent p1=new Child();
        p1.m1();
        //Child c1=new Parent(); //invalid
    }
```

Inheritance

Inheritance is a mechanism where we will derive a class in the presence of existing class.

or

Inheritance is a mechanism where one class will inherit the properties of another class.

We have five types of inheritance.

1)Single level inheritance

2)Multi level inheritance

3)Multiple inheritance

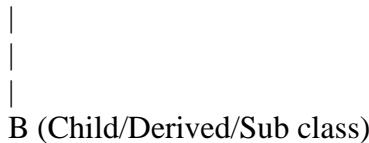
4)Hierarchical inheritance

5)Hybrid inheritance

1)Single level inheritance

If we derived a class in the presence of one base class is called single level inheritance.

Diagram: A (Parent/Base/Super class)



```
class A
{
    public void m1()
    {
        System.out.println("M1-Method");
    }
}
class B extends A
{
    public void m2()
    {
        System.out.println("M2-Method");
    }
}
class Test
{
    public static void main(String[] args)
    {
        A a=new A();
        a.m1();
        B b=new B();
        b.m1();
        b.m2();
    }
}
```

2)Multi level inheritance

If we derived a class in the presence of one base class and that class will derived from another base class is called multi level inheritance.

Diagram: A



```
class A
{
```

```

public void m1()
{
    System.out.println("M1-Method");
}
}
class B extends A
{
    public void m2()
    {
        System.out.println("M2-Method");
    }
}
class C extends B
{
    public void m3()
    {
        System.out.println("M3-Method");
    }
}
class Test
{
    public static void main(String[] args)
    {
        A a=new A();
        a.m1();
        B b=new B();
        b.m1();
        b.m2();
        C c=new C();
        c.m1();
        c.m2();
        c.m3();
    }
}

```

3)Multiple inheritance

In java a class can't extends more then one class simultenously because java does not support multiple inheritance.

ex: class A
{
 }
 class B
{
 }
 class C extends A,B --> Invalid
{
}

But interface can extends more then one interface simultaneously so we can achieve multiple inheritance concept through interfaces.

ex: interface A

```
{}  
interface B
```

```
{}  
interface C extends A,B --> valid
```

```
{}
```

If our class does not extends any other class then our class is a direct child class of Object class.

ex: Diag:

```
class A          Object  
{             |  
}             |  
              A
```

If our class extends some other class then our class is an indirect child class of Object class.

ex: Diag:

```
class A          Object  
{             |  
}             |  
class B extends A      A  
{             |  
}             |  
              B
```

Java does not support cyclic inheritance.

ex: class A extends B

```
{  
}  
class B extends A  
{  
}
```

Q) Why java does not support multiple inheritance?

There may chance of raising ambiguity problem that's why java does not support multiple inheritance.

ex: p1.m1() p2.m1()
 |-----|
 | |
 c.m1()

4) Hierarchical inheritance

If we derived multiple classes in the presence of one base class is called hierarchical inheritance.

Diagram:
 A
 |
 |-----|
 B C

class A

```
{
```

```

public void m1()
{
    System.out.println("M1-Method");
}
class B extends A
{
    public void m2()
    {
        System.out.println("M2-Method");
    }
}
class C extends A
{
    public void m3()
    {
        System.out.println("M3-Method");
    }
}
class Test
{
    public static void main(String[] args)
    {
        A a=new A();
        a.m1();

        B b=new B();
        b.m1();
        b.m2();

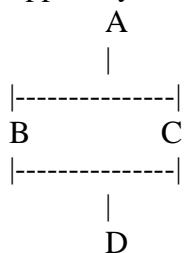
        C c=new C();
        c.m1();
        c.m3();
    }
}

```

5)Hybrid inheritance

Hybrid inheritance is a combination of more than one inheritance.
Java does not support hybrid inheritance.

Diagram:



Has-A relationship

Has-A relationship is also known as Composition and Aggregation.

There is no specific keyword to implements Has-A relationship but mostly we will use new operator.

The main objective of Has-A relationship is to provide reusability.

Has-A relationship will increase dependency between two components.

ex: class Engine

```
{  
    -  
        - //engine specific functionality  
    -  
}  
class Car  
{  
    Engine e=new Engine();  
}
```

class Ihub

```
{  
    public String courseName()  
    {  
        return "Full Stack Java With AWS";  
    }  
    public double courseFee()  
    {  
        return 30000d;  
    }  
    public String trainerName()  
    {  
        return "Niyaz Sir";  
    }  
}  
class Usha  
{  
    public void getCourseDetails()  
    {  
        Ihub i=new Ihub();  
        System.out.println("Course Name :" +i.courseName());  
        System.out.println("Course Fee :" +i.courseFee());  
        System.out.println("Trainer Name :" +i.trainerName());  
    }  
}  
class Student  
{  
    public static void main(String[] args)
```

```

{
    Usha u=new Usha();
    u.getCourseDetails();
}
}

```

Composition

Without existing container object there is no chance of having contained object then the relationship between container and contained object is called composition which is strongly association.

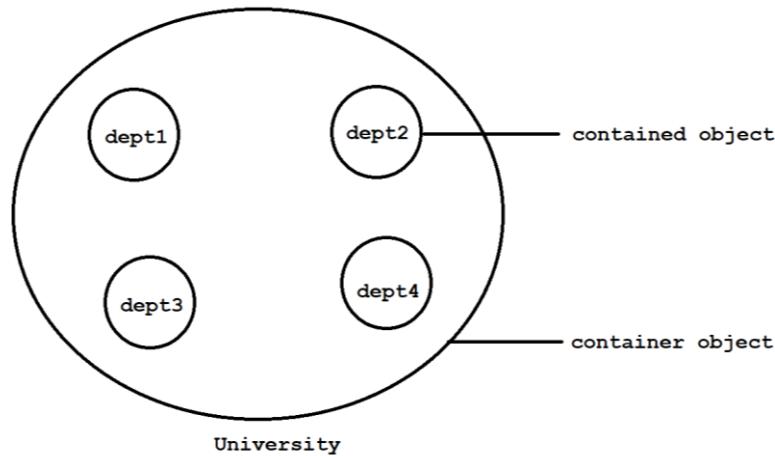


Diagram: java24.1

Aggregation

Without existing container object there is a chance of having contained object then the relationship between container and contained object is called aggregation which is loosely association.

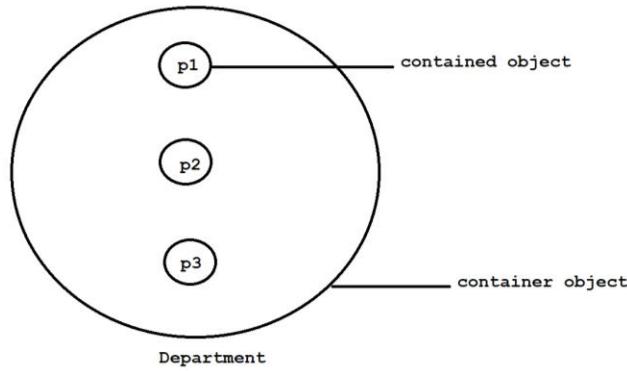


Diagram: java24.2

Method overloading

Having same method name with different parameters in a single class is class method overloading.

All the methods present in a class are called overloaded methods.

Method overloading will reduce complexity of the programming.

```

class MeeSeva
{
    //overloaded methods
    public void search(int voterId)
    {
        System.out.println("Details found using voterId");
    }
    public void search(String houseNo)
    {
        System.out.println("Details found using house No");
    }
    public void search(long aadharNo)
    {
        System.out.println("Details found using aadhar No ");
    }
}
class Test
{
    public static void main(String[] args)
    {
        MeeSeva ms=new MeeSeva();
        ms.search(101);
        ms.search("1-6-4/1/A");
        ms.search(200021);
    }
}

```

Method overriding

Having same method name with same parameters in a two different classes is called method overriding.

Methods which are present in parent class are called overridden methods.

Methods which are present in child class are called overriding methods.

```

class Parent
{
    //overridden methods
    public void property()
    {
        System.out.println("Cash+Gold+Land");
    }
    public void marry()
    {
        System.out.println("Subhalakshmi");
    }
}
class Child extends Parent
{

```

```

//overriding methods
public void marry()
{
    System.out.println("Rashmika");
}
}

class Test
{
    public static void main(String[] args)
    {
        Parent p=new Parent();
        p.property(); // Cash+Gold+Land
        p.marry(); //subhalakshmi

        Child c=new Child();
        c.property(); // Cash+Gold+Land
        c.marry(); // Rashmika

        Parent p1=new Child();
        p1.property(); // Cash+Gold+Land
        p1.marry(); // Rashmika
    }
}

```

If we declare any method as final then overriding is not possible.

ex:

```

class Parent
{
    //overridden methods
    public void property()
    {
        System.out.println("Cash+Gold+Land");
    }

    public final void marry()
    {
        System.out.println("Subhalakshmi");
    }
}

class Child extends Parent
{
    //overriding methods
    public void marry()
    {
        System.out.println("Rashmika");
    }
}

```

```

class Test
{
    public static void main(String[] args)
    {
        Parent p=new Parent();
        p.property(); // Cash+Gold+Land
        p.marry(); //subhalakshmi

        Child c=new Child();
        c.property(); // Cash+Gold+Land
        c.marry(); // Rashmika

        Parent p1=new Child();
        p1.property(); // Cash+Gold+Land
        p1.marry(); // Rashmika
    }
}

```

Method Hiding

Method hiding is exactly same as method overriding with following differences.

Method overriding

All the methods present in method overriding must be non-static.

Method resolution will taken care by a JVM based on runtime object.

It is also known as Runtime polymorphism, Dynamic polymorphism, late binding.

Method Hiding

All the methods present in method Hiding must be static.

Method resolution will taken care by compiler based on reference type.

It is also known as compile time polymorphism, Static polymorphism, early binding.

```

class Parent
{
    //overridden methods
    public static void property()
    {
        System.out.println("Cash+Gold+Land");
    }
    public static void marry()
    {
        System.out.println("Subhalakshmi");
    }
}
class Child extends Parent
{

```

```

//overriding methods
public static void marry()
{
    System.out.println("Rashmika");
}
}

class Test
{
    public static void main(String[] args)
    {
        Parent p=new Parent();
        p.property(); // Cash+Gold+Land
        p.marry(); //subhalakshmi

        Child c=new Child();
        c.property(); // Cash+Gold+Land
        c.marry(); // Rashmika

        Parent p1=new Child();
        p1.property(); // Cash+Gold+Land
        p1.marry(); // Subhalakshmi
    }
}

```

Interview Questions

Q) Can we overload main method in java?

Yes, we can overload main method in java but JVM always execute main method with String[] parameter only.

```

class Test
{
    public static void main(int[] iargs)
    {
        System.out.println("int argument");
    }

    public static void main(String[] args)
    {
        System.out.println("String argument");
    }
}

```

Q) Can we override main method in java?

No , We can't override main method in java because it is static.

Polymorphism

Polymorphism has taken from Greek word.

Here poly means many and morphism means forms.

The ability to represent in different forms is called polymorphism.

The main objective of polymorphism is to provide flexibility.

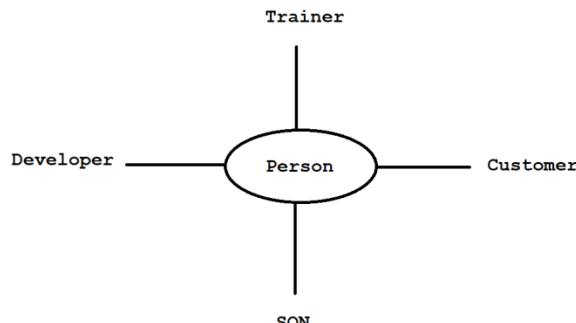


Diagram: java25.1

In java polymorphism is divided into two types.

- 1)Compile time polymorphism / static polymorphism / early binding
- 2)Runtime polymorphism / dynamic polymorphism / late binding

1)Compile time polymorphism

A polymorphism which exhibits at compile time is called compile time polymorphism.

- ex: Method Overloading
 Method Hiding

2)Runtime polymorphism

A polymorphism which exhibits at runtime is called runtime polymorphism.

- ex: Method overriding

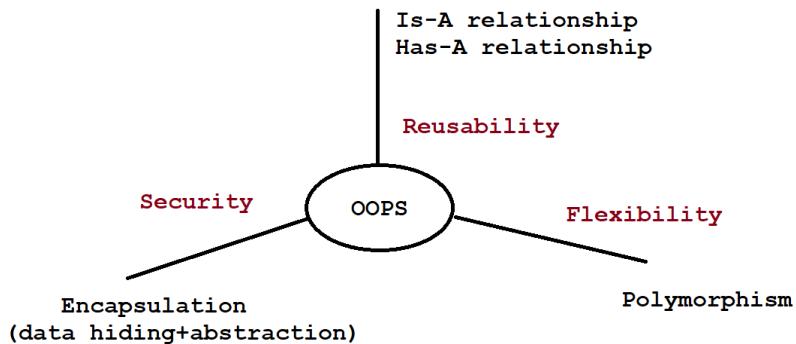


Diagram: java25.2

Constructors

A constructor is a special method which is used to initialized an object.

Having same name as class name is called constructor.

A constructor does not allow any return type.

A constructor will execute when we create an object.

A constructor will accept following modifiers.

- ex: default, public, private & protected

In java constructors are divided into two types.

- 1)Userdefined constructor
- 2)Default constructor

1) Userdefined constructor

A constructor which is created by the user based on the application requirement is called userdefined constructor.

It is classified into two types.

- i) Zero-Argument constructor
- ii) Parameterized constructor
- i) Zero-Argument constructor

Suppose if we won't pass any argument to userdefined constructor then that constructor is called zero-argument constructor.

```
class Test
{
    Test()
    {
        System.out.println("0-arg const");
    }
    public static void main(String[] args)
    {
        System.out.println("Main-Method");
    }
}
```

o/p: Main-Method

```
class Test
{
    Test()
    {
        System.out.println("0-arg const");
    }
    public static void main(String[] args)
    {
        System.out.println("Main-Method");
        Test t=new Test();
    }
}
```

o/p: Main-Method
0-arg const

```
class Test
{
    public Test()
    {
        System.out.println("0-arg const");
    }
    public static void main(String[] args)
    {
```

```

        Test t1=new Test();
        System.out.println("Main-Method");
        Test t2=new Test();
    }
}
o/p: 0-arg const
      Main-Method
      0-arg const

```

ii) Parameterized constructor

Suppose if we are passing atleast one argument to userdefined constructor then that constructor is called parameterized constructor.

```

class Employee
{
    private int empId;
    private String empName;
    private double empSal;

    //parameterized constructor
    public Employee(int empId,String empName,double empSal)
    {
        this.empId=empId;
        this.empName=empName;
        this.empSal=empSal;
    }
    public void getEmployeeDetails()
    {
        System.out.println("Employee Id :"+empId);
        System.out.println("Employee Name :"+empName);
        System.out.println("Employee Salary :"+empSal);
    }
}
class Test
{
    public static void main(String[] args)
    {
        Employee e=new Employee(101,"Alan Morries",1000d);
        e.getEmployeeDetails();
    }
}

```

2) **Default constructor**

It is a compiler generated constructor for every java program where we are not defining atleast zero argument constructor.

Default constructor is a empty implementation.

We can see default constructor by using below command.

```
ex: javap -c Test
```

```
class Test
{
    public static void main(String[] args)
    {
        System.out.println("Hello Java World");
    }
}

class Test
{
    //default constructor
    Test();
    public static void main(String[] args)
    {
        System.out.println("Hello Java World");
    }
}
```



Diagram: java25.3

this keyword

A this keyword is a java keyword which is used to refer current class object reference.

We can utilize this keyword in following ways.

- i) To refer current class variables
- ii) To refer current class methods
- iii) To refer current class constructors

i) To refer current class variables

```
class A
{
    int i=10;
    int j=20;
    A(int i,int j)
    {
        System.out.println(i+" "+j); // 100 200
        System.out.println(this.i+" "+this.j); //10  20
    }
}

class Test
{
    public static void main(String[] args)
    {
        A a=new A(100,200);
    }
}
```

ii) To refer current class methods

```
class A
{
    public void m1()
    {
        System.out.println("M1 method");
        this.m2();
    }

    public void m2()
    {
        System.out.println("M2 method");
    }
}
```

```

        }
    }
class Test
{
    public static void main(String[] args)
    {
        A a=new A();
        a.m1();
    }
}

```

iii) To refer current class constructor

```

class A
{
    A()
    {
        System.out.println("0-arg const");
    }
    A(int i)
    {
        this();
        System.out.println("int arg const");
    }
    A(double d)
    {
        this(10);
        System.out.println("double arg const");
    }
}
class Test
{
    public static void main(String[] args)
    {
        A a=new A(10.5d);
    }
}

```

super keyword

A super keyword is a java keyword which is used to refer super class object reference.
We can utilize super keyword in following ways.

- i) To refer super class variables
- ii) To refer super class methods
- iii) To refer super class constructors

i) To refer super class variables

```

class A

```

```

{
    int i=10;
    int j=20;
}
class B extends A
{
    int i=100;
    int j=200;
    B(int i,int j)
    {
        System.out.println(this.i+" "+this.j); // 100 200
        System.out.println(super.i+" "+super.j); //10 20
        System.out.println(i+" "+j); // 1000 2000
    }
}
class Test
{
    public static void main(String[] args)
    {
        B b=new B(1000,2000);
    }
}

```

ii) To refer super class methods

```

class A
{
    public void m1()
    {
        System.out.println("M1 method");
    }
}
class B extends A
{
    public void m2()
    {
        super.m1();
        System.out.println("M2 method");
    }
}
class Test
{
    public static void main(String[] args)
    {
        B b=new B();
        b.m2();
    }
}
```

```
}
```

iii) To refer super class constructors

```
class A
{
    A()
    {
        System.out.println("A const");
    }
}

class B extends A
{
    B()
    {
        super();
        System.out.println("B const");
    }
}

class Test
{
    public static void main(String[] args)
    {
        new B();
    }
}
```

Interfaces

An interface is a collection of zero or more abstract methods.

Abstract methods are incomplete methods because they ends with semicolon and do not have any body.

It is not possible to create object for interfaces.

To write the implementation of abstract methods of an interface we will use implementation class.

It is possible to create object for implementation class.

By default every abstract method is a public and abstract.

Interface contains only constants i.e public static final.

syntax: interface <interface_name>

```
{
    -
    - //abstract methods
    - //constants
    -
}
```

If we know Service Requirement Specification then we need to use interface.

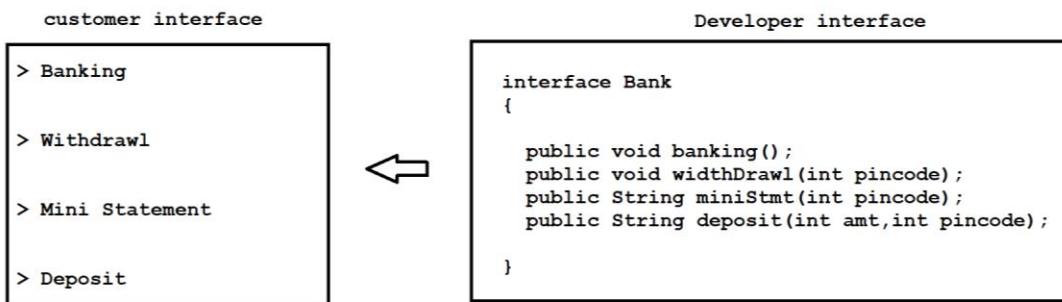


Diagram: java26.1

interface A

```

{
    //abstract method
    public abstract void m1();
}

class B implements A
{
    public void m1()
    {
        System.out.println("M1 method");
    }
}

class Test
{
    public static void main(String[] args)
    {
        A a=new B();
        a.m1();
    }
}

```

ex:

```

interface A
{
    //abstract method
    public abstract void m1();
}

class Test
{
    public static void main(String[] args)
    {
        //Anonymous inner class
        A a=new A()
        {
            public void m1()

```

```

        {
            System.out.println("M1 method");
        }
    };
    a.m1();
}
}

```

If interface contains four methods then we need to override all methods otherwise we will get compile time error.

```

interface A
{
    //abstract methods
    public abstract void show();
    public void display();
    abstract void view();
    void see();
}

class B implements A
{
    public void show()
    {
        System.out.println("show-method");
    }
    public void display()
    {
        System.out.println("display-method");
    }
    public void view()
    {
        System.out.println("view-method");
    }
    public void see()
    {
        System.out.println("see-method");
    }
}

class Test
{
    public static void main(String[] args)
    {
        A a=new B();
        a.show();
        a.display();
    }
}

```

```
        a.see();
        a.view();
    }
}
```

In java, a class can't extends more then one class but interface can extends more then one interface.

interface A

```
{  
    void m1();  
}
```

interface B

```
{  
    void m2();  
}
```

interface C extends A,B

```
{  
    void m3();  
}
```

class D implements C

```
{  
    public void m1()  
    {  
        System.out.println("M1-method");  
    }  
    public void m2()  
    {  
        System.out.println("M2-method");  
    }  
    public void m3()  
    {  
        System.out.println("M3-method");  
    }  
}
```

class Test

```
{  
    C c =new D();  
    c.m1();  
    public static void main(String[] args)  
    {  
        c.m2();  
        c.m3();  
    }  
}
```

```

A class can implements more then one interface.

interface Father
{
    //constant
    float HT=6.2f;
    void height();
}

interface Mother
{
    float HT=5.8f;
    void height();
}

class Child implements Father,Mother
{
    public void height()
    {
        float height=(Father.HT+Mother.HT)/2;
        System.out.println("Child Height :" +height);
    }
}

class Test
{
    public static void main(String[] args)
    {
        Child c=new Child();
        c.height();
    }
}

```

Java 8

According java 8 version, interface is a collection of default methods, static methods and abstract methods.

Marker interface

Interface which does not have any constants or abstract methods is called marker interface.

Emprty interface is called marker interface.

Using marker interface we will get some ability to do.

We have following list of marker interface.

ex: Serializable
 Cloneable
 Remote
 and etc.

Abstract classes

Abstract class is a collection of zero or more abstract methods and concrete methods.

A abstract keyword is applicable for methods and classes but not for variables.

It is not possible to create object for abstract class.

To write the implementation of abstract methods of a abstract class we will use sub classes.
By default every abstract method is a public and abstract.

Abstract class contains only instance variables.

syntax: abstract class class_name

```
{  
    -  
    - // instance variables  
    - // abstract methods  
    - // concrete methods  
    -  
}
```

If we know partial implementation then we need to use Abstract class.

abstract class Plan

```
{  
    //instance variable  
    protected double rate;  
  
    //abstract method  
    public abstract void getRate();  
  
    //concrete method  
    public void calculateBillAmt(int units)  
    {  
        System.out.println("Total Units :" + units);  
        System.out.println("Total Bill :" + rate * units);  
    }  
}  
class DomesticPlan extends Plan  
{  
    public void getRate()  
    {  
        rate=2.5d;  
    }  
}  
class CommercialPlan extends Plan  
{  
    public void getRate()  
    {  
        rate=5.0d;  
    }  
}  
class Test  
{  
    public static void main(String[] args)  
    {  
        DomesticPlan dp=new DomesticPlan();  
    }  
}
```

```

        dp.getRate();
        dp.calculateBillAmt(250);

        CommercialPlan cp=new CommercialPlan();
        cp.getRate();
        cp.calculateBillAmt(250);
    }
}

```

Q)What is the difference between interface and abstract class?

<u>interface</u>	<u>abstract class</u>
To declare interface we will use interface keyword.	To declare abstract class we wil use abstract keyword.
Interface is a collection of abstract methods,default methods and static methods.	It is a collection of abstract methods and concrete methods.
It does not allow constructor.	It allows constructor.
It does not allow blocks.	It allows blocks.
To write the implementation of abstract methods we will use implementation class.	To write the implementation of abstract methods we will use sub classes.
It contains only constants.	It contains only instance variables.
Multiple inheritance is possible.	Multiple inheritance is not possible.
If we know only specification then we need to use interface.	If we know partial implementation then we need to use abstract class.

API

API stands for Application Programming Interface.

It is a base for the programmer to develop software applications.

API is a collection of packages.

We have three types of API's.

1)Predefined API

Built-In API is called predefined API.

ex: <https://docs.oracle.com/javase/8/docs/api/>

2)Userdefined API

API which is created by the user based on the application requirement.

3)Third party API

API which is given by third party vendor.

ex: JAVAZOOM API

iText API

and etc.

Packages

Package is a collection of classes ,interfaces , enums and Annotations.

Here enum is a special class and Annotation is a special interface.

In general, a package is a collection of classes and interfaces.

Package is also known as folder or a directory.

In java, we have two types of packages.

1)Predefined packages

2)Userdefined packages

1)Predefined packages

Built-In packages are called predefined packages.

ex: java.lang

 java.io

 java.util

 java.text

 java.util.stream

 java.sql

 javax.servlet

and etc.

2)Userdefined packages

Packages which are created by the user based on the application requirement are called userdefined packages.

We can declare userdefined package as follow.

syntax: package <package_name>;

package com.ihub.www;

import java.util.Calendar;

class Test

{

 public static void main(String[] args)

 {

 Calendar c=Calendar.getInstance();

 //convert time to 24 hours

 int h=c.get(Calendar.HOUR_OF_DAY);

 if(h<12)

 System.out.println("Good Morning");

 else if(h<16)

 System.out.println("Good Afternoon");

 else if(h<20)

 System.out.println("Good Evening");

 else

 System.out.println("Good Night");

 }

}

We can compile the code by using below command.

ex: current directory

 |
javaprog> javac -d . Test.java

 |
 Destination folder

We can run the code by using below command.

ex: javaproj> java com.ihub.www.Test
| |
pkg name classname

Singleton class

A class which allows us to create only one object is called singleton class.

Using a class if we call any method and that method returns same class object is called singleton class.

We have following list of singleton class.

ex: Calendar
LocalTime
LocalDate
and etc.

To create our own singleton class we need to use private constructor and factory method.

class Singleton

{

```
    static Singleton singleton=null;  
  
    //private constructor  
    private Singleton()  
    {  
  
    }  
    //factory method  
    public static Singleton getInstance()  
    {  
        if(singleton==null)  
        {  
            singleton=new Singleton();  
        }  
        return singleton;  
    }  
}
```

class Test

{

```
    public static void main(String[] args)  
    {  
        Singleton s1=Singleton.getInstance();  
        System.out.println(s1.hashCode());  
  
        Singleton s2=Singleton.getInstance();  
        System.out.println(s2.hashCode());  
    }  
}
```

Inner classes

Sometimes we will declare a class inside another class such concept is called inner class.

ex: class <Outer_Class>

```
{  
    class <Inner_Class>  
    {  
        - //code to be declare  
        -  
    }  
}
```

Inner classes introduced as a part of event handling to remove GUI bugs.

Because of powerful features and benefits of inner classes, programmer started to use inner class in our regular programming.

In inner class , we can't declare static members.

Accessing inner class data from static area of outer class

```
class Outer  
{  
    class Inner  
    {  
        public void m1()  
        {  
            System.out.println("M1-Method");  
        }  
    }  
    public static void main(String[] args)  
    {  
        Outer.Inner i=new Outer().new Inner();  
        i.m1();  
    }  
}
```

Note:

If we compile above program we will get two .class files i.e
Outer.class and Outer\$Inner.class

ex:

```
class Outer  
{  
    class Inner  
    {  
        public void m1()  
        {  
            System.out.println("M1-Method");  
        }  
    }  
}
```

```

public static void main(String[] args)
{
    new Outer().new Inner().m1();
}
}

```

Accessing inner class data from non-static area of outer class

```

class Outer
{
    class Inner
    {
        public void m1()
        {
            System.out.println("M1-Method");
        }
    }
    public void m2()
    {
        Inner i=new Inner();
        i.m1();
    }
    public static void main(String[] args)
    {
        Outer o=new Outer();
        o.m2();
    }
}

```

Enum

Enum is a group of named constants.

Enum concept introduced in 1.5v.

Using enum we can create our own datatype called enumerated datatype.

When compare to old language enum , java enum is more powerful.

syntax: enum type_name

```

{
    val1,val2,..valN
}

```

ex: enum Months

```

{
    JAN,FEB,MAR
}

```

Internal implementation of enum

Every enum internally implements as final class concept and extends with java.lang.Enum class.
Every enum constant is a reference variable of enum type.

```

enum Months      public final class Months extends java.lang.Enum
{
{
JAN,FEB,MAR => public static final Months JAN=new Months();
}
public static final Months FEB=new Months();
public static final Months MAR=new Months();
}

```

Declaration and Usage of enum

```

enum Months
{
    JAN,FEB,MAR
}
class Test
{
    public static void main(String[] args)
    {
        Months m=Months.JAN;
        System.out.println(m);//JAN
    }
}
ex:
enum Months
{
    JAN,FEB,MAR
}

class Test
{
    public static void main(String[] args)
    {
        Months m=Months.FEB;
        switch(m)
        {
            case JAN: System.out.println("January"); break;
            case FEB: System.out.println("February"); break;
            case MAR: System.out.println("March"); break;
        }
    }
}

```

java.lang.Enum class

Power to enum will be inherited from java.lang.Enum class.

It contains following two methods.

- 1)values() It is a static method which returns group of constants from enum.
- 2)ordinal() It returns ordinal number of enum.

```

enum Week
{
    MON,TUE,WED,THU,FRI,SAT,SUN
}

class Test
{
    public static void main(String[] args)
    {
        Week[] w=Week.values();

        //for each loop
        for(Week w1:w)
        {
            System.out.println(w1+" ----- "+w1.ordinal());
        }
    }
}

```

When compare to old language enum , java enum is more powerful because in addition to constants we can declare variables, constructors and methods.

```

enum Cloths
{
    SILK,KHADI,COTTON;

    Cloths()
    {
        System.out.println("constructor");
    }
}

```

```

class Test
{
    public static void main(String[] args)
    {
        Cloths c=Cloths.SILK;
    }
}

```

ex:

```

enum Cloths
{
    SILK,KHADI,COTTON;

    static int i=10;
}

```

```

public static void main(String[] args)
{
    System.out.println(i);//10
}
}

```

Wrapper classes

The main objective of wrapper classes are

- 1) To wrap primitive type to wrapper object and vice versa.
- 2) To define several utility methods.

<u>Primitive type</u>	<u>Wrapper class</u>
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

constructor

Every wrapper class contains two constructors. One will take corresponding primitive as an argument and another will take corresponding String as an argument.

<u>Wrapper class</u>	<u>constructor</u>
Byte	byte or String
Short	short or String
Integer	int or String
Long	long or String
Float	float or String
Double	double or String
Boolean	boolean or String
Character	char

```

class Test
{
    public static void main(String[] args)
    {
        Integer i1=new Integer(10);
        System.out.println(i1);//10

        Integer i2=new Integer("20");
        System.out.println(i2);//20
    }
}

```

```

class Test
{
    public static void main(String[] args)
    {
        Boolean b1=new Boolean(true);
        System.out.println(b1);//true

        Boolean b2=new Boolean("false");
        System.out.println(b2);//false
    }
}

```

ex:

```

class Test
{
    public static void main(String[] args)
    {
        Character c=new Character('a');
        System.out.println(c); //a
    }
}

```

Utility methods

1) valueOf()

It is similar to constructor.

It is used to convert primitive type to wrapper object.

```

class Test
{
    public static void main(String[] args)
    {
        Integer i1=Integer.valueOf(10);
        System.out.println(i1);//10

        Long l1=Long.valueOf("20");
        System.out.println(l1);//20
    }
}

```

2) xxxValue()

It will convert wrapper object to primitive type.

```

class Test
{
    public static void main(String[] args)
    {
        Integer i=new Integer(10);

```

```
    byte b=i.byteValue();
    System.out.println(b);

    short s=i.shortValue();
    System.out.println(s);
}
}
```

3) parseXxx()

It is used to convert string type to primitive type.

```
class Test
{
    public static void main(String[] args)
    {
        String str="100";

        int i=Integer.parseInt(str);
        System.out.println(i); //100

        long l=Long.parseLong(str);
        System.out.println(l); //100

        float f=Float.parseFloat(str);
        System.out.println(f); //100.0
    }
}
```

4) toString()

It is used to convert wrapper object type to String type.

```
class Test
{
    public static void main(String[] args)
    {
        Integer i1=new Integer(10);

        String s=i1.toString();

        System.out.println(s);
    }
}
```

Q) Write a java program to perform sum of two binary numbers?

input: 1010

 0101

output: 1111

ex:

```

import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the first binary number :");
        String binary1=sc.next(); //1010

        System.out.println("Enter the second binary number :");
        String binary2=sc.next(); //0101

        //converting binary to decimal
        int a=Integer.parseInt(binary1,2);
        int b=Integer.parseInt(binary2,2);

        int c=a+b;

        //converting decimal to binary
        String result=Integer.toBinaryString(c);
        System.out.println("sum of two binary numbers is =" +result);
    }
}

```

Q) Types of objects in java?

We have two types of objects in java.

1) Immutable object

2) Mutable object

1) Immutable object

After object creation if we perform any changes then for every change a new object will be created such type of object is called immutable object.

ex: String and Wrapper classes

2) Mutable object

After object creation if we perform any changes then all the required changes will be done in a same object such type of object is called mutable object.

ex: StringBuffer and StringBuilder

String

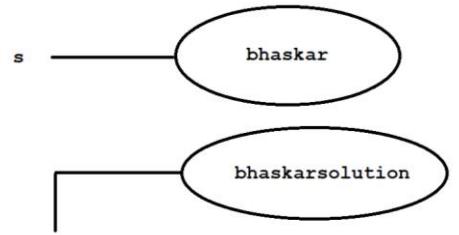
String is a set of characters which is enclosed in a double quotation.

String is an immutable object.

case1: After object creation we can't perform any changes. If we perform any changes then for every change a new object will be created such behaviour is called immutability of an object.

ex: String s=new String("bhaskar");
s.concat("solution");

```
System.out.println(s); // bhaskar
```



No reference then it is eligible for Garbage Collector

Diagram: java28.1

case2:

Q)What is the difference between == and .equals() method?

==

It is a equality operator or comparision operator which always returns boolean value.
It is used to reference comparision or address comparision.

class Test

```
{  
    public static void main(String[] args)  
    {  
        String s1=new String("bhaskar");  
        String s2=new String("bhaskar");  
  
        System.out.println(s1==s2);//false  
    }  
}
```

.equals()

It is a method present in String class which always returns boolean value.
It is used for content comparision which is case sensitive.

class Test

```
{  
    public static void main(String[] args)  
    {  
        String s1=new String("bhaskar");  
        String s2=new String("bhaskar");  
  
        System.out.println(s1.equals(s2));//true  
    }  
}
```

case3: Once if we create a String object, two objects will be created.One is on heap and another is on SCP (String Constant Pool) area.But 's' always points to heap area only.

```
String s=new String("bhaskar");
```

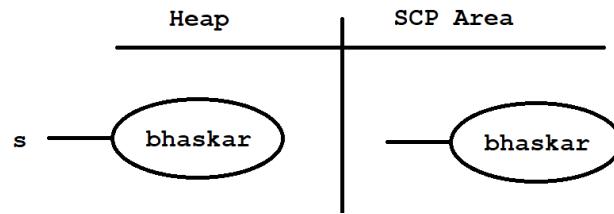


Diagram: java28.2

Object creation in SCP area is always optional. First JVM Will check is there any object is created with same content or not. If it is created then JVM simply refers to that object. If it is not created then JVM will create a new object. Hence there is no chance of having duplicate objects in SCP area.

Even though SCP object do not have any reference, Garbage collector can't access them. SCP objects will destroy when JVM shutdowns or terminated.

```
String s1=new String("bhaskar");
String s2=new String("bhaskar");
String s3="bhaskar";
String s4="bhaskar";
String s5="solution";
```

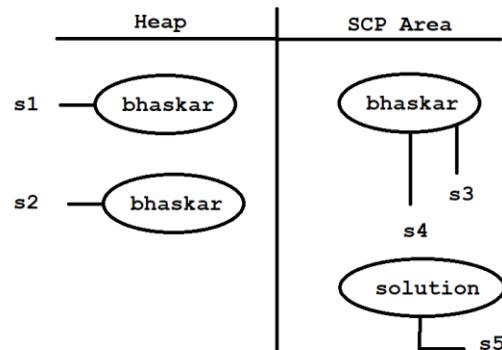


Diagram: java29.1

Interning of String object

With the help of heap object reference if we want corresponding SCP object reference then we need to use intern() method.

```
String s1=new String("bhaskar");
String s2=s1.intern();
String s3="bhaskar";
System.out.println(s2==s3); // true
```

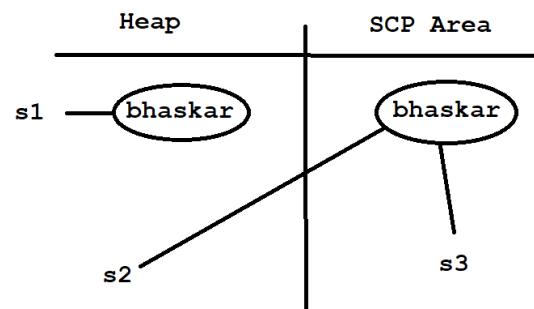


Diagram: java29.2

String important methods

Q) Write a java program to accept one string and display it?

```
import java.util.Scanner;
```

```
class Test
```

```
{
```

```
public static void main(String[] args)
{
    Scanner sc=new Scanner(System.in);

    System.out.println("Enter the string :");
    String str=sc.next();

    System.out.println(str);
}
```

```
approach2
import java.util.Scanner;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the string :");
        String str=sc.nextLine();

        System.out.println(str);
    }
}
```

Q)Write a java program to remove all special characters from given string?

input: Ihub@Talent_M\$angeme#nt515;

output: IhubTalentManageent515

class Test

```
{
    public static void main(String[] args)
    {
        String str="Ihub@Talent_M$angeme#nt515";

        str=str.replaceAll("[^A-Za-z0-9]","");
        System.out.println(str);
    }
}
```

Q)Write a java program to find out length of the string?

input: hello

output: 5

class Test

```
{
```

```
public static void main(String[] args)
{
    String str="hello";
    System.out.println(str.length());//5
}
```

Q)Write a java program to convert uppercase letter to lowercase letter?

input: IHUBTALENT

output: ihubtalent

class Test

```
{
    public static void main(String[] args)
    {
        String str="IHUBTALENT";
        str=str.toLowerCase();
        System.out.println(str);//ihubtalent
    }
}
```

Q)Write a java program to convert lowercase letter to uppercase letter?

input: ihubtalent

output: IHUBTALENT

class Test

```
{
    public static void main(String[] args)
    {
        String str="ihubtalent";
        str=str.toUpperCase();
        System.out.println(str);//IHUBTALENT
    }
}
```

Q)Write a java program to find out given the strings are equal or not?

input: str1 = "ihub";

str2 = "ihub";

output: Both are equals

class Test

```
{
    public static void main(String[] args)
    {
```

```

String str1="ihub";
String str2="ihub";

if(str1.equals(str2))
    System.out.println("Both are equals");
else
    System.out.println("Both are not equal");
}

ex:
class Test
{
    public static void main(String[] args)
    {
        String str1="ihub";
        String str2="IHUB";

        if(str1.equalsIgnoreCase(str2))
            System.out.println("Both are equals");
        else
            System.out.println("Both are not equal");
    }
}

```

Q)Write a java program to check given word present in a string or not?

input: str = "Welcome to ihub talent management"

word = "ihub"

output: It is available

class Test

{

```

public static void main(String[] args)
{
    String str="Welcome to ihub talent management";

    String word="ihub";

    if(str.contains(word))
        System.out.println("It is available");
    else
        System.out.println("It is not available");
}

```

Q)Write a java program to display the character from string which is present in given index?

input: str = "ihubtalent"

```

index = 5
output: a
class Test
{
    public static void main(String[] args)
    {
        String str="ihubtalent";

        int index =5;

        System.out.println(str.charAt(index));//a
    }
}

```

Q)Write a java program to find out index of first occurrence character in a given string?

input: str = "ihubtalentmanagement"

ch='a';

output: 5

class Test

```

{
    public static void main(String[] args)
    {
        String str = "ihubtalentmanagement";

        char ch='a';

        System.out.println(str.indexOf(ch));
    }
}

```

Q)Write a java program to find out index of last occurrence character in a given string?

input: str = "ihubtalentmanagement"

ch='a';

output: 13

class Test

```

{
    public static void main(String[] args)
    {
        String str = "ihubtalentmanagement";

        char ch='a';

        System.out.println(str.lastIndexOf(ch));
    }
}

```

Q)Write a java program to find out number of occurrence of a given character?

```
input: str = "ihubtalentmanagement"
      char ch='a';
output: 3 times
class Test
{
    public static void main(String[] args)
    {
        String str = "ihubtalentmanagement";
        char ch='a';

        int cnt=0;
        for(int i=0;i<str.length();i++)
        {
            char alphabet=str.charAt(i);

            if(alphabet==ch)
            {
                cnt++;
            }
        }

        System.out.println(cnt+" times");
    }
}
```

Q)Write a java program to display the string in reverse order?

```
input: hello
output: olleh
class Test
{
    public static void main(String[] args)
    {
        String str="hello";

        char[] carr=str.toCharArray();

        String rev="";
        for(int i=carr.length-1;i>=0;i--)
        {
            rev+=carr[i];
        }
        System.out.println(rev);//olleh
    }
}
```

Q)Write a java program to check given string is palindrome or not?

input: madam

output: It is palindrome string

class Test

{

 public static void main(String[] args)

 {

 String str="madam";

 char[] carr=str.toCharArray();

 String rev="";

 for(int i=carr.length-1;i>=0;i--)

 {

 rev+=carr[i];

 }

 if(str.equals(rev))

 System.out.println("It is palindrome string");

 else

 System.out.println("It is not palindrome string");

 }

}

Q)Write a java program to display reverse of a sentence?

input: This Is Java Class

output: Class Java Is This

class Test

{

 public static void main(String[] args)

 {

 String str="This Is Java Class";

 String[] sarr=str.split(" ");

 String rev="";

 for(int i=sarr.length-1;i>=0;i--)

 {

 rev+=sarr[i]+" ";

 }

 System.out.println(rev);

 }

}

Q)Write a java program to display reverse of a word in a sentence?

input: This Is Java Class

output: sihT sI avaJ ssalC

class Test

{

 public static void main(String[] args)

 {

 String str="This Is Java Class";

 String[] sarr=str.split(" "); // This Is Java Class

 //for each loop

 String rev="";

 for(String s:sarr)

 {

 char[] carr=s.toCharArray();

 for(int i=carr.length-1;i>=0;i--)

 {

 rev+=carr[i];

 }

 //add space

 rev+=" ";

 }

 System.out.println(rev);

 }

}

Q)Write a java program to find out duplicate characters from given string?

input: google

output: go

class Test

{

 public static void main(String[] args)

 {

 String str="google";

 String characters="";

 String duplicates="";

 for(int i=0;i<str.length();i++)

 {

 String current=Character.toString(str.charAt(i));

```

        if(characters.contains(current))
        {
            if(!duplicates.contains(current))
            {
                duplicates+=current;
                continue;
            }
            characters+=current;
        }

        System.out.println(duplicates);
    }
}

```

Q)Write a java program to find out unique/distinct characters from given string?

input: google

output: gole

class Test

```

{
    public static void main(String[] args)
    {
        String str="google";

        String characters="";
        String duplicates="";

        for(int i=0;i<str.length();i++)
        {
            String current=Character.toString(str.charAt(i));

            if(characters.contains(current))
            {
                if(!duplicates.contains(current))
                {
                    duplicates+=current;
                    continue;
                }
            }
            characters+=current;
        }

        System.out.println(characters);
    }
}

```

Assignments

Q)Write a java program to number of uppercase letters, lowercase letters, digits , words and spaces?

input: This Is Java Class 29

output: uppercase : 4

lowercase : 11

digits : 2

words : 5

spaces : 4

Q)Write a java program to display the string starting with uppecase letter?

input: This is Java class for Freshers

output: This Java Freshers

Q)Write a java program to display the strings which are palindrome?

input: racar is madam for dad

output: racar madam dad

Q)Write a java program to display the string below format?

input: A1B2C3D4

output: ABBCCCDDDD

class Test

{

 public static void main(String[] args)

 {

 String str="A1B2C3D4";

 for(int i=0;i<str.length();i++)

 {

 if(Character.isAlphabetic(str.charAt(i)))

 {

 System.out.print(str.charAt(i));

 }

 else

 {

 int a=Character.getNumericValue(str.charAt(i)); // 1 2 3 4

 for(int j=1;j<a;j++)

 {

 System.out.print(str.charAt(i-1));

 }

 }

 }

 }

}

Q)Write a java program to find most repeating character in a given string?

input: googleformat

output: o is repeating for 3 times

class Test

{

 public static void main(String[] args)

 {

 String str="googleformat";

 int maxCount=0;

 char element=' ';

 for(int i=0;i<str.length();i++)

 {

 int cnt=0;

 for(int j=0;j<str.length();j++)

 {

 if(str.charAt(i)==str.charAt(j))

 {

 cnt++;

 }

 }

 if(maxCount<cnt)

 {

 maxCount=cnt;

 element=str.charAt(i);

 }

 }

 System.out.println(element+" is repeating for "+maxCount+" times");

 }

}

Q)Write a java program to perform right rotation of a string?

input: str = "ihubtalent";

 cnt = 2;

output: ubtalentih

class Test

{

 public static void main(String[] args)

 {

 String str="ihubtalent";

 int cnt=2;

```

        char[] carr=str.toCharArray();           // i h u b t a l e n t
        String str1=str.substring(cnt,carr.length); //ubtalent
        String str2=str.substring(0,cnt); //ih
        System.out.println(str1+str2); //ubtalentih
    }
}

```

Q)Write a java program to find out left rotation of a string?

input: str="ihubtalent"

cnt=2;

output: ntihubtale

class Test

{

public static void main(String[] args)

{

String str="ihubtalent";

int cnt=2;

char[] carr=str.toCharArray(); // i h u b t a l e n t

String str1=str.substring(0,carr.length-cnt); //ihubtale

String str2=str.substring(carr.length-cnt,carr.length); //nt

System.out.println(str2+str1);

}

}

Q)Write a java program to display the string starting with vowels?

input: this is java class for you in a coming batch

output: is in a

class Test

{

public static void main(String[] args)

{

String str="this is java class for you in a coming batch";

String[] sarr=str.split(" ");

String result="";

```

        for(String s:sarr)
    {
        if(s.charAt(0)=='a' || s.charAt(0)=='e' || s.charAt(0)=='i' ||
s.charAt(0)=='o' || s.charAt(0)=='u')
        {
            result+=s+" ";
        }
    }
    System.out.println(result);
}
}

```

Assignments

Q)Write a java program to display given string in below format?

input: ABBCCCDDDD

output: A1B2C3D4

Q)Write a java program to display permutation of string?

input: ABC

output: ABC

ACB

BAC

BCA

CAB

CBA

StringBuffer

If our content change frequently then it is never recommended to use String object because for every change a new object will be created.

To overcome this limitation Sun Micro System introduced StringBuffer object.

In StringBuffer object all the required changes will be done in a single object only and it is mutable object.

constructor

1) StringBuffer sb=new StringBuffer()

It will create empty StringBuffer object with default initial capacity of 16.

If capacity reaches to maximum capacity then new capacity will be created with below formula.

ex: new capacity = current_capacity+1*2;

class Test

{

 public static void main(String[] args)

 {

 StringBuffer sb=new StringBuffer();

 System.out.println(sb.capacity());//16

```

        sb.append("abcdefghijklnop");
        System.out.println(sb.capacity());//16
        sb.append("qr");// 16+1*2=34
        System.out.println(sb.capacity());//34
    }
}

```

2) StringBuffer sb=new StringBuffer(int initialcapacity)

It will create StringBuffer with specified initial capacity.

```

class Test
{
    public static void main(String[] args)
    {
        StringBuffer sb=new StringBuffer(19);
        System.out.println(sb.capacity());//19
    }
}

```

3)StringBuffer sb=new StringBuffer(String str)

It will create StringBuffer object which is equivalent to String.

Here capacity will be created with below formulae.

ex: capacity = s.length+16;

```

class Test
{
    public static void main(String[] args)
    {
        StringBuffer sb=new StringBuffer("bhaskar");
        System.out.println(sb.capacity());//7+16=23
    }
}

```

Q)Write a java program to display reverse of a string?

input: hello

output: olleh

class Test

```

{
    public static void main(String[] args)
    {

```

```

        String str="hello";
        String rev="";
        StringBuffer sb=new StringBuffer(str);
        rev=sb.reverse().toString();
        System.out.println(rev);
    }
}

```

Q)Write a java program to check given string is palindrome or not?

```

class Test
{
    public static void main(String[] args)
    {
        String str="madam";
        String rev="";
        StringBuffer sb=new StringBuffer(str);
        rev=sb.reverse().toString();
        if(str.equals(rev))
            System.out.println("It is palindrome string");
        else
            System.out.println("It is not palindrome string");
    }
}

```

StringBuilder

StringBuilder is exactly the same as StringBuffer with following differences.

<u>StringBuffer</u>	<u>StringBuilder</u>
All the methods present in StringBuffer are synchronized.	No method present in StringBuilder is synchronized.

At a time only one thread is allowed to execute.Hence we can achieve thread safety.

Multiple threads are allowed to execute.Hence we can't achieve thread safety.

Waiting time of a thread will increase effectively performance is low.

There is no waiting time of a thread effectively performance is high.

It is introduced in 1.0v.

It is introduced in 1.5v.

StringTokenizer

A StringTokenizer is a class which is present in java.util package.

It is used to tokenize the string irrespective of regular expression.

We can create StringTokenizer object as follow.

syntax: StringTokenizer st=new StringTokenizer(String,RegEx);

StringTokenizer class contains following methods.

ex: public boolean hasMoreTokens()
public String nextToken()
public boolean hasMoreElements()
public Object nextElement()
public int countTokens()

import java.util.StringTokenizer;

class Test

```
{  
    public static void main(String[] args)  
    {  
        StringTokenizer st=new StringTokenizer("this is java class");  
  
        System.out.println(st.countTokens());//4  
    }  
}
```

*Note: Here default regular expression is space.

import java.util.StringTokenizer;

class Test

```
{  
    public static void main(String[] args)  
    {  
        StringTokenizer st=new StringTokenizer("this is java class"," ");  
  
        System.out.println(st.countTokens());//4  
    }  
}
```

ex:

import java.util.StringTokenizer;

class Test

```
{  
    public static void main(String[] args)  
    {  
        StringTokenizer st=new StringTokenizer("this is java class"," ");  
  
        while(st.hasMoreTokens())  
        {  
            String s=st.nextToken();  
            System.out.println(s);  
        }  
    }  
}
```

```
        }
    }

ex:
import java.util.StringTokenizer;
class Test
{
    public static void main(String[] args)
    {
        StringTokenizer st=new StringTokenizer("this is java class","");
        while(st.hasMoreElements())
        {
            String s=(String)st.nextElement();
            System.out.println(s);
        }
    }
}
```

```
ex:
import java.util.StringTokenizer;
class Test
{
    public static void main(String[] args)
    {
        StringTokenizer st=new StringTokenizer("9,99,999","","");
        while(st.hasMoreElements())
        {
            String s=(String)st.nextElement();
            System.out.println(s);
        }
    }
}
```

java.io package

File

```
File f=new File("abc.txt");
File will check is there any abc.txt file already created or not.
If it is available it simply refers to that file.If it is not created then
it won't create any new file.
import java.io.*;
class Test
{
    public static void main(String[] args)
    {
```

```
        File f=new File("abc.txt");
        System.out.println(f.exists());//false
    }
}
```

A File object can be used to create a physical file.

```
import java.io.*;
class Test
{
    public static void main(String[] args)throws IOException
    {
        File f=new File("abc.txt");
        System.out.println(f.exists());//false

        f.createNewFile();
        System.out.println(f.exists());//true

    }
}
```

A File object can be used to create a directory also.

```
import java.io.*;
class Test
{
    public static void main(String[] args)throws IOException
    {
        File f=new File("bhaskar123");
        System.out.println(f.exists());//false

        f.mkdir();
        System.out.println(f.exists());//true

    }
}
```

Q)Write a java program to Create a "cricket123" folder and inside that folder create "abc.txt" file?

```
import java.io.*;
class Test
{
    public static void main(String[] args)throws IOException
    {
        File f1=new File("cricket123");
        f1.mkdir();

        File f2=new File("cricket123","abc.txt");
```

```

f2.createNewFile();

System.out.println("Please check the location");

}

}

```

FileWriter

FileWriter is used to write character oriented data into a file.

constructor

```

FileWriter fw=new FileWriter(String s);
FileWriter fw=new FileWriter(File f);
ex:   FileWriter fw=new FileWriter("aaa.txt");
      or

```

```

File f=new File("aaa.txt");
FileWriter fw=new FileWriter(f);

```

If file does not exist then FileWriter will create a physical file.

Methods

- | | |
|---------------------|--|
| 1. write(int ch) | It will insert single character into a file. |
| 2. write(char[] ch) | It will insert array of characters into a file. |
| 3. write(String s) | It will insert String into a file. |
| 4. flush() | It gives guarantee that last character of a file is also inserted. |
| 5. close() | It is used to close the FileWriter object. |

```

import java.io.*;
class Test
{
    public static void main(String[] args) throws IOException
    {
        FileWriter fw=new FileWriter("aaa.txt");

        fw.write(98);// b
        fw.write("\n");

        char[] ch={'a','b','c'};
        fw.write(ch);
        fw.write("\n");

        fw.write("bhaskar\nsolution");
        fw.flush();
        fw.close();
        System.out.println("Please check the location");
    }
}

```

FileReader

It is used to read character oriented data from a file.

constructor

```
FileReader fr=new FileReader(String s);
FileReader fr=new FileReader(File f);
Eg:   FileReader fr=new FileReader("aaa.txt");
      or
      File f=new File("aaa.txt");
      FileReader fr=new FileReader(f);
```

Methods

- | | |
|--------------------|---|
| 1. read() | It will read next character from a file and return unicode value.
If next character is not available then it will return -1. |
| 2. read(char[] ch) | It will read collection of characters from a file. |
| 3. close() | It is used to close FileReader object. |

```
import java.io.*;
class Test
{
    public static void main(String[] args) throws IOException
    {
        FileReader fr=new FileReader("aaa.txt");
    }
}
```

```
int i=fr.read();
while(i!=-1)
{
    System.out.print((char)i);
    i=fr.read();
}
fr.close();
```

ex:2

```
import java.io.*;
class Test
{
    public static void main(String[] args) throws IOException
    {
        FileReader fr=new FileReader("aaa.txt");
        char[] carr=new char[255];
        //load the data from file to char array
        fr.read(carr);
        //reading the data from char array
        for(char c:carr)
```

```

    {
        System.out.print(c);
    }

    fr.close();

}
}

```

Usage of FileWriter and FileReader is not recommended to use

While inserting the data by using FileWriter ,we need to insert line seperator(\n) which is very headache for the programmer.

While reading the data by using FileReader object ,we need to read character by character which is not convenient to the programmer.

To overcome this limitation Sun micro system introduced BufferedWriter and BufferedReader.

BufferedWriter

It is used to insert character oriented data into a file.

constructor

```
BufferedWriter bw=new BufferedWriter(Writer w);
BufferedWriter bw=new BufferedWriter(Writer w,int buffersize);
```

BufferedWriter object does not communicate with files directly.

It will take the support of some writer objects.

```
ex:   FileWriter fw=new FileWriter("bbb.txt");
      BufferedWriter bw=new BufferedWriter(fw);
           or
      BufferedWriter bw=new BufferedWriter(new FileWriter("bbb.txt"));
```

Methods

- | | |
|---------------------|--|
| 1. write(int ch) | It will insert single character into a file. |
| 2. write(char[] ch) | It will insert array of characters into a file. |
| 3. write(String s) | It will insert String into a file. |
| 4. flush() | It gives guarantee that last character of a file is also inserted. |
| 5. close() | It is used to close the BufferedWriter object. |
| 6. newLine() | It will insert new line into a file. |

```
import java.io.*;
class Test
{
    public static void main(String[] args)throws IOException
    {
```

```
        BufferedWriter bw=new BufferedWriter(new FileWriter("bbb.txt"));
        bw.write(98);//b
        bw.newLine();
        char[] ch={'a','b','c'};
        bw.write(ch);
```

```

        bw.newLine();

        bw.write("bhaskar");
        bw.newLine();

        bw.flush();
        bw.close();
        System.out.println("Please check the location");
    }
}

```

BufferedReader

It is enhanced reader to read character oriented data from a file.

constructor

```

BufferedReader br=new BufferedReader(Reader r);
BufferedReader br=new BufferedReader(Reader r,int buffersize);

```

BufferedReader object can't communicate with files directly. IT will take support of some reader objects.

```

ex:   FileReader fr=new FileReader("bbb.txt");
      BufferedReader br=new BufferedReader(fr);
      or
      BufferedReader br=new BufferedReader(new FileReader("bbb.txt"));

```

The main advantage of BufferedReader over FileReader is we can read character line by line instead of character by character.

methods

- | | |
|--------------------|---|
| 1. read() | It will read next character from a file and return unicode value.
If next character is not available then it will return -1. |
| 2. read(char[] ch) | It will read collection of characters from a file. |
| 3. close() | It is used to close BufferedReader object. |
| 4. nextLine() | It is used to read next line from the file. If next line is not then it available will return null. |

```

import java.io.*;
class Test
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader br=new BufferedReader(new FileReader("bbb.txt"));
        String line=br.readLine();
        while(line!=null)
        {
            System.out.println(line);
            line=br.readLine();
        }
        br.close();
    }
}

```

PrintWriter

It is enhanced write to write character oriented data into a file.

constructor

PrintWriter pw=new PrintWriter(String s);

PrintWriter pw=new PrintWriter(File f);

PrintWriter pw=new PrintWriter(Writer w);

PrintWriter can communicate with files directly and it will take the support of some writer objects.

ex: PrintWriter pw=new PrintWriter("ccc.txt");

 or

 PrintWriter pw=new PrintWriter(new File("ccc.txt"));

 or

 PrintWriter pw=new PrintWriter(new FileWriter("ccc.txt"));

The main advantage of PrintWriter over FileWriter and BufferedWriter is we can insert any type of data.

Assume if we want insert primitive values then PrintWriter is best choice.

methods

write(int ch)

write(char[] ch)

write(String s)

flush()

close()

writeln(int i)

writeln(float f)

writeln(double d)

writeln(String s)

writeln(char c)

writeln(boolean b)

write(int i)

write(float f)

write(double d)

write(String s)

write(char c)

write(boolean b)

ex:

```
import java.io.*;
```

```
class Test
```

```
{
```

```
    public static void main(String[] args) throws IOException
```

```
{
```

```
        PrintWriter pw=new PrintWriter("ccc.txt");
```

```
        pw.write(100);// d
```

```

        pw.println(100);// 100
        pw.print('a');
        pw.println(true);
        pw.println("hi");
        pw.println(10.5d);

        pw.flush();
        pw.close();
        System.out.println("Please check the location");
    }
}

```

Various ways to provide input values from keyboard

There are many ways to provide input values from keyboard.

- 1) command line argument
- 2) Console class
- 3) BufferedReader class
- 4) Scanner class

- 1) command line argument

```

class Test
{
    public static void main(String[] args)
    {
        String name=args[0];

        System.out.println("Welcome :"+name);
    }
}

```

o/p:

```
javac Test.java
```

```
java Test DennisRitchie
```

- 2) Console class

```

import java.io.*;
class Test
{
    public static void main(String[] args)throws IOException
    {
        Console c=System.console();

        System.out.println("Enter the name :");

        String name=c.readLine();
    }
}

```

```

        System.out.println("Welcome :" + name);
    }
}

3) BufferedReader class
import java.io.*;
class Test
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("Enter the name :");

        String name=br.readLine();

        System.out.println("Welcome :" + name);
    }
}

4) Scanner class
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the employee id :");
        int id=sc.nextInt();

        System.out.println("Enter the employee name :");
        String name=sc.next();

        System.out.println("Enter the employee salary :");
        float sal=sc.nextFloat();

        System.out.println(id+" "+name+" "+sal);
    }
}

```

Exception Handling

Q)What is the difference between Exception and Error?

Exception

Exception is a problem for which we can provide solution programmatically.

Exception will raise due to syntax error.

ex: FileNotFoundException
 ArithmetricException
 IllegalArgumentException

Error

Error is a problem for which we can't provide solution programmatically.

Error will raise due to lack of system resources.

ex: OutOfMemoryError
 LinkageError
 StackOverFlowError
 and etc

As a part of java application development, it is a responsibility of a programmer to provide smooth termination for every java program.

We have two types of terminations.

- 1)Smooth termination / graceful termination
- 2)Abnormal termination

1)Smooth termination

During the program execution suppose if we are not getting any interruption in the middle of the program such type of termination is called smooth termination.

2)Abnormal termination

During the program execution suppose if we are getting any interruption in the middle of the program such type of termination is called abnormal termination

class Test

```
{  
    public static void main(String[] args)  
    {  
        System.out.println(10/0);  
    }  
}
```

If any exception raised in our program , we must and should handle that exception otherwise our program will terminates abnormally.

Exception will display name of the exception ,description of the exception and line number of the exception.

Exception

It is a unwanted, unexpected event which disturbs normal flow of our program.

Exceptions always raised at runtime so they are also known as runtime events.

The main objective of exception handing is to provide graceful termination.

In java, exceptions are divided into two types.

- 1)Predefined exception

2) Userdefined exceptions

1) Predefined exception

Built-In exceptions are called predefined exceptions.

It classified into two types.

i) Checked exceptions

ii) Unchecked exceptions

i) Checked exceptions

Exceptions which are checked by the compiler at the time of compilation are called checked exceptions.

ex: InterruptException
 IOException
 EOFException
 and etc.

ii) Unchecked exceptions

Exceptions which are checked by the JVM at the time of runtime are called unchecked exceptions.

ex: ArithmaticException
 IllegalArgumentException
 ClassCastException
 and etc.

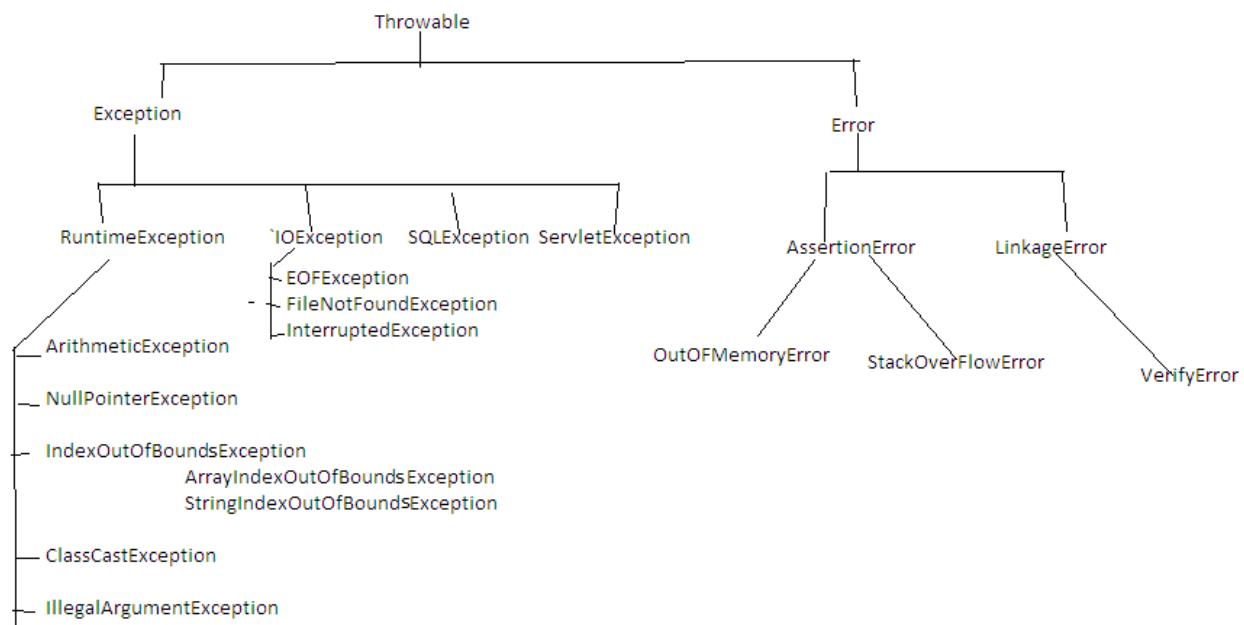


Diagram: java32.1

If any checked exception raised in our program , we must and should handle that exception by using try and catch block.

try block

It is a block which contains risky code.

A try block associate with catch block.

If any exception raised in try block then it won't be executed.
A try block is used to throw the exception to catch block.

catch block

It is a block which contains Error Handling Code.

A catch block always associate with try block.

A catch block will take exception name as a parameter and that name must match with exception class name.

If there is no exception in try block then catch block won't be executed.

A catch block is used to catch the exception which is thrown by try block.

syntax:

```
try
{
    -
    - //Risky Code
    -
}

catch(ArithmeticException ae)
{
    -
    - //Error Handling Code
    -
}

class Test
{
    public static void main(String[] args)
    {
        try
        {
            System.out.println("try-block");
        }
        catch (Exception e)
        {
            System.out.println("catch-block");
        }
    }
}
```

o/p: try-block

ex:

```
class Test
{
    public static void main(String[] args)
    {
        try
        {
```

```
        System.out.println(10/0);
    }
    catch (ArithmaticException ae)
    {
        System.out.println("catch-block");
    }
}
o/p:  catch-block
```

ex:

```
class Test
{
    public static void main(String[] args)
    {
        try
        {
            System.out.println("stmt1");
            System.out.println(10/0);
            System.out.println("stmt2");
        }
        catch (ArithmaticException ae)
        {
            System.out.println("catch-block");
        }
    }
}
o/p:  stmt1
      catch-block
```

A try with multiple catch blocks

A try block can have multiple catch blocks.

If a try block contains multiple catch block then order of catch block is very important ,it should be from child to parent but not from parent to child.

```
class Test
{
    public static void main(String[] args)
    {
        try
        {
            System.out.println(10/0);
        }
        catch (ArithmaticException ae)
        {
            System.out.println("From AE");
        }
    }
}
```

```

        catch (RuntimeException re)
        {
            System.out.println("From RE");
        }
        catch (Exception e)
        {
            System.out.println("From E");
        }
    }
}

```

Various ways to display exception details

Throwable class defined following three methods to display exception details.

1) printStackTrace()

It will display name of the exception, description of the exception and line number of the exception.

2) toString()

It will display name of the exception and description of the exception.

3) getMessage()

It will display description of the exception.

```

class Test
{
    public static void main(String[] args)
    {
        try
        {
            System.out.println(10/0);
        }
        catch (ArithmaticException ae)
        {
            ae.printStackTrace();

            System.out.println("=====");
            System.out.println(ae.toString());
            System.out.println("=====");

            System.out.println(ae.getMessage());
        }
    }
}

```

A single catch block can handle multiple exceptions.

ex:

```
class Test
```

```

{
    public static void main(String[] args)
    {
        try
        {
            System.out.println(10/0);
        }
        catch (ArithmaticException | ClassCastException | IllegalArgumentException e)
        {
            e.printStackTrace();
        }
    }
}

```

finally block

It is never recommended to maintain cleanup code in try block because if any exception raised in try block then it won't be executed.

It is never recommended to maintain cleanup code in catch block because if no exception raised in try block then catch block won't be executed.

We need a place where we can maintain cleanup code and it will execute irrespective of exception raised or not. Such block is called finally block.

A finally block always associate with try and catch block.

syntax:

```

try
{
    -
    - //Risky Code
    -
}

catch(ArithmaticException ae)
{
    -
    - //Error Handling Code
    -
}

finally
{
    -
    - //Cleanup Code
    -
}

```

ex:

```

class Test
{
    public static void main(String[] args)
    {
        try

```

```
{  
    System.out.println("try-block");  
}  
catch (ArithmaticException ae)  
{  
    ae.printStackTrace();  
}  
finally  
{  
    System.out.println("finally-block");  
}  
}  
}
```

o/p: try-block
finally-block

ex:

```
class Test  
{  
    public static void main(String[] args)  
    {  
        try  
        {  
            System.out.println(10/0);  
        }  
        catch (ArithmaticException ae)  
        {  
            ae.printStackTrace();  
        }  
        finally  
        {  
            System.out.println("finally-block");  
        }  
    }  
}
```

o/p: java.lang.ArithmaticException: / by zero
at Test.main(Test.java:7)
finally-block

ex:

```
import java.io.*;  
class Test  
{  
    public static void main(String[] args)  
    {  
        FileWriter fw=null;  
  
        try
```

```

{
    fw=new FileWriter("xyz.txt");
    fw.write(98);//b
    fw.write("\n");
    char[] carr={'a','b','c'};
    fw.write(carr);
    fw.flush();
}
catch (IOException ioe)
{
    ioe.printStackTrace();
}
finally
{
    System.out.println("Please check the location");

    try
    {
        fw.close();
    }
    catch (IOException ioe)
    {
        ioe.printStackTrace();
    }
}
}
}

```

A try with finally combination is valid in java.

```

class Test
{
    public static void main(String[] args)
    {
        try
        {
            System.out.println("try-block");
        }
        finally
        {
            System.out.println("finally-block");
        }
    }
}

```

Q)What is the difference between final, finally and finalized method?

final

A final is a modifier which is applicable for variables,methods and classes.

If we declare any variable as final then reassignment of that variable is not possible.

If we declare any method as final then overriding of that method is not possible.

If we declare any class as final then creating child class is not possible.

finally

It is a block which contains cleanup code and it will execute irrespective of exception raised or not.

finalized method

It is a method called by garbage collector just before destroying an object for cleanup activity.

throw statement

Sometimes we will create exception object explicitly and handover to JVM manually by using throw statement.

ex: throw new ArithmeticException("Don't divide by zero");

```
class Test
{
    public static void main(String[] args)
    {
        throw new ArithmeticException("don't divide by zero");
    }
}
```

throws statement

If any checked exception raised in our program so we must and should handle that exception by using try and catch block or by using throws statement.

```
class Test
{
    public static void main(String[] args)
    {
        try
        {
            Thread.sleep(3000);
            System.out.println("Welcome to Java");
        }
        catch (InterruptedException ie)
        {
            ie.printStackTrace();
        }
    }
}
```

```

ex:
class Test
{
    public static void main(String[] args) throws InterruptedException
    {
        Thread.sleep(5000);
        System.out.println("Welcome to Java");
    }
}

```

2) Userdefined exceptions

Exceptions which are created by the user based on the application requirement are called userdefined exceptions.

ex: NoInterestInPracticeException
 EnjoyingClassesWithMobileException
 ACNotWorkingException
 TooYoungException
 TooOldException
 and etc.

```

import java.util.Scanner;
class TooYoungException extends RuntimeException
{
    TooYoungException(String s)
    {
        super(s);
    }
}
class TooOldException extends RuntimeException
{
    TooOldException(String s)
    {
        super(s);
    }
}
class Test
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the age :");
        int age=sc.nextInt();
        if(age<18)
            throw new TooYoungException("U r not eligible to vote");
        else
    }
}

```

```
        throw new TooOldException("User is eligible to vote");
    }
}
```

Generics

Arrays are typesafe.

It means we can provide guarantee that what type of elements are present in arrays.

If requirement is there to store string values then it is recommended to use String[] array.

```
ex:   String[] sarr=new String[10];
      sarr[0]="hi";
      sarr[1]="hello";
      sarr[2]="bye";
      sarr[3]=10; --> invalid
```

At the time of retrieving the data from array we don't need to perform any typecasting.

```
ex:   String[] sarr=new String[10];
      sarr[0]="hi";
      sarr[1]="hello";
      sarr[2]="bye";
      -
```

```
      -
```

```
      String val=sarr[0];
```

Collections are not typesafe.

We can't give guarantee that what type of elements are present in Collections.

If requirement is there to store String values then it is never recommended to use ArrayList because we won't get any compile time error or runtime error but sometimes our program will get failure.

```
ex:   ArrayList al=new ArrayList();
      al.add("hi");
      al.add("hello");
      al.add("bye");
      al.add(10);
```

At the time of retrieving the data from Collections ,compulsory we need to perform typecasting.

```
ex:   ArrayList al=new ArrayList();
```

```
      al.add("hi");
      al.add("hello");
      al.add("bye");
      al.add(10);
      -
```

```
      -
```

```
      String val=(String)al.get(0);
```

To overcome above limitations Sun Micro System introduced Generics concept in 1.5 version.

The main objective of generics are.

- 1) To make Collections as typesafe.
- 2) To avoid typecasting problem.

java.util package

Q)What is the difference between Arrays and Collections ?

Arrays	Collections
It is a collection of homogeneous data elements.	It is a collection of homogeneous and heterogeneous data elements.
Arrays are fixed in size.	Collections are growable in nature.
Performance point of view arrays are recommended to use.	Memory point of view Collections are recommended to use.
It is typesafe.	It is not typesafe.
Arrays can hold primitive types and object types.	Collections can hold only object types
Arrays are not implemented based on data structure concept so we can't expect any ready made methods. For every logic we need to write the code explicitly.	Collections are implemented based on data structure concept so we can expect ready made methods.

Collection

Collection is an interface which is present in java.util package.

It is a root interface for entire Collection framework.

If we want to represent group of individual objects in a single entity then we need to use Collection interface.

Collection interface contains following methods which are applicable for entire Collection objects.

ex: cmd> javap java.util.Collection

```
public abstract int size();
public abstract boolean isEmpty();
public abstract boolean contains(java.lang.Object);
public abstract java.util.Iterator<E> iterator();
public abstract java.lang.Object[] toArray();
public abstract <T> T[] toArray(T[]);
public abstract boolean add(E);
public abstract boolean remove(java.lang.Object);
public abstract boolean containsAll(java.util.Collection<?>);
public abstract boolean addAll(java.util.Collection<? extends E>);
public abstract boolean removeAll(java.util.Collection<?>);
public boolean removeIf(java.util.function.Predicate<? super E>);
public abstract boolean retainAll(java.util.Collection<?>);
public abstract void clear();
public abstract boolean equals(java.lang.Object);
public abstract int hashCode();
public java.util.Spliterator<E> spliterator();
public java.util.stream.Stream<E> stream();
public java.util.stream.Stream<E> parallelStream();
```

List

It is a child interface of Collection interface.

If we want to represent group of individual objects in a single entity where duplicate objects are allowed and order is preserved then we need to use List interface.

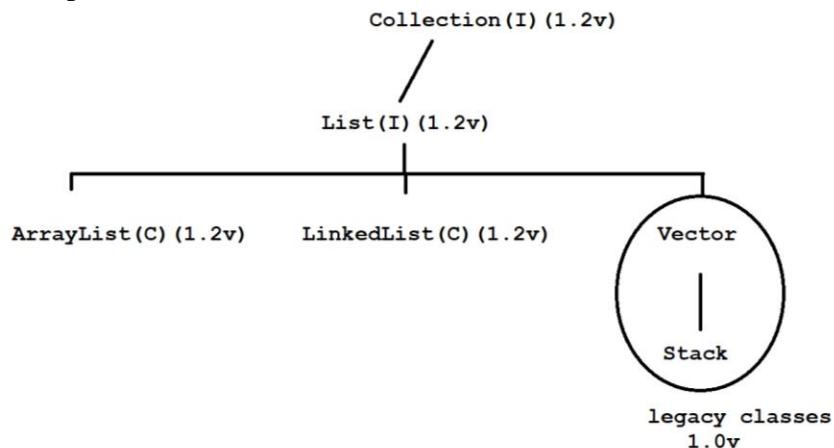


Diagram : java33.1

ArrayList

The underlying data structure is resizable array or growable array.

Duplicate objects are allowed.

Insertion order is preserved.

Heterogeneous objects are allowed.

Null insertion is possible.

It implements Serializable, Cloneable and RandomAccess interface.

If our frequent operation is a retrieval operation then `ArrayList` is best choice.

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        ArrayList al=new ArrayList();
        al.add("one");
        al.add("two");
        al.add("three");
        System.out.println(al);//[one,two,three]
        al.add("one");
        System.out.println(al);//[one,two,three,one]
        al.add(10);
        System.out.println(al);//[one,two,three,one,10]
        al.add(null);
        System.out.println(al);//[one,two,three,one,10,null]
    }
}
```

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        ArrayList<String> al=new ArrayList<String>();
        al.add("one");
        al.add("two");
        al.add("three");
        System.out.println(al);//[one,two,three]
        al.add("one");
        System.out.println(al);//[one,two,three,one]
        al.add(null);
        System.out.println(al);//[one,two,three,one,null]
    }
}
```

ex:

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        ArrayList<String> al=new ArrayList<String>();
        al.add("one");
        al.add("two");
        al.add("three");

        for(int i=0;i<al.size();i++)
        {
            String s=al.get(i);
            System.out.println(s);
        }
    }
}
```

ex:

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        ArrayList<String> al=new ArrayList<String>();
        al.add("one");
        al.add("two");
        al.add("three");
```

```
System.out.println(al.isEmpty());//false  
System.out.println(al.contains("one"));//true  
  
al.remove("one");  
System.out.println(al);//[two,three]  
  
al.clear();  
System.out.println(al);//[]  
}  
}
```

ex:

```
import java.util.*;  
class Test  
{  
    public static void main(String[] args)  
    {  
        List<String> list=new ArrayList<String>();  
        list.add("one");  
        list.add("two");  
        list.add("three");  
        System.out.println(list);//[one,two,three]  
    }  
}
```

ex:

```
import java.util.*;  
class Test  
{  
    public static void main(String[] args)  
    {  
        List<String> list=new ArrayList<String>();  
        list.add(new String("one"));  
        list.add(new String("two"));  
        list.add(new String("three"));  
        System.out.println(list);//[one,two,three]  
    }  
}
```

ex:

```
import java.util.*;  
class Test  
{  
    public static void main(String[] args)
```

```

    {
        List<String> l=Arrays.asList("one","two","three","four");
        System.out.println(l);
    }
}

```

LinkedList

The underlying data structure is double linkedlist.

Duplicate objects are allowed.

Order is preserved.

Hetrogeneous objects are allowed.

Null insertion is possible.

It implements Serializable, Cloneable and Deque interface.

If our frequent operation is a adding and removing in the middle then LinkedList is a best choice.

LinkedList contains following methods.

ex: addFirst()

addLast()

getFirst()

getLast()

removeFirst()

removeLast()

and etc.

import java.util.*;

class Test

{

 public static void main(String[] args)

 {

 LinkedList ll=new LinkedList();

 ll.add("one");

 ll.add("two");

 ll.add("three");

 System.out.println(ll);//[one,two,three]

 ll.add("one");

 System.out.println(ll);//[one,two,three,one]

 ll.add(10);

 System.out.println(ll);//[one,two,three,one,10]

 ll.add(null);

 System.out.println(ll);//[one,two,three,one,10,null]

 }

}

ex:

import java.util.*;

class Test

{

 public static void main(String[] args)

```

{
    LinkedList<String> ll=new LinkedList<String>();
    ll.add("one");
    ll.add("two");
    ll.add("three");
    System.out.println(ll);//[one,two,three]
    ll.addFirst("gogo");
    ll.addLast("jojo");
    System.out.println(ll);//[gogo,one,two,three,jojo]

    System.out.println(ll.getFirst());//gogo
    System.out.println(ll.getLast());//jojo

    ll.removeFirst();
    ll.removeLast();
    System.out.println(ll);//[one,two,three]
}
}

```

ex:

```

import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        LinkedList<String> ll1=new LinkedList<String>();
        ll1.add("one");
        ll1.add("two");
        ll1.add("three");
        System.out.println(ll1);//[one,two,three]

        LinkedList<String> ll2=new LinkedList<String>();
        ll2.add("raja");
        System.out.println(ll2);//[raja]

        ll2.addAll(ll1);
        System.out.println(ll2);//[raja,one,two,three]

        System.out.println(ll2.containsAll(ll1));// true

        ll2.removeAll(ll1);
        System.out.println(ll2);//[raja]
    }
}

```

Vector

The underlying data structure is resizable array or growable array.

Duplicate objects are allowed.

Insertion order is preserved.

Hetrogeneous objects are allowed.

Null insertion is possible.

It implements Serializable, Cloneable and RandomAccess interface.

All the methods present in Vector are synchronized.Hence we can achieve thread safety.

Vector contains following methods.

ex: addElement()
 firstElement()
 lastElement()
 removeElementAt()
 removeAllElements()
 and etc.

```
import java.util.*;  
class Test  
{  
    public static void main(String[] args)  
    {  
        Vector v=new Vector();  
        v.add("one");  
        v.add("two");  
        v.add("three");  
        System.out.println(v);//[one,two,three]  
        v.add("one");  
        System.out.println(v);//[one,two,three,one]  
        v.add(10);  
        System.out.println(v);//[one,two,three,one,10]  
        v.add(null);  
        System.out.println(v);//[one,two,three,one,10,null]  
  
    }  
}
```

ex:

```
import java.util.*;  
class Test  
{  
    public static void main(String[] args)  
    {  
        Vector v=new Vector();  
        v.addElement("one");  
        v.addElement("two");  
        v.addElement("three");  
        System.out.println(v);//[one,two,three]
```

```

        System.out.println(v.firstElement());//one
        System.out.println(v.lastElement());//three

        v.removeElementAt(1);
        System.out.println(v);//[one,three]

        v.removeAllElements();
        System.out.println(v);//[]

    }

}


```

ex:

```

import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        Vector v=new Vector();
        v.add("one");
        v.add("two");
        v.add("three");
        System.out.println(v);//[one,two,three]

        System.out.println(v.get(0));//one
        System.out.println(v.get(v.size()-1));//three

        v.remove("two");
        System.out.println(v);//[one,three]

        v.clear();
        System.out.println(v);//[]

    }
}

```

Stack

It is a child class of Vector class.

If we depends upon Last In First Out (LIFO) order then we need to use Stack.

We can create Stack object as follow.

ex: Stack s =new Stack();

Methods

- | | |
|-------------------|--|
| 1) push(Object o) | It will insert the element to stack. |
| 2) pop() | It will remove the element from stack. |
| 3) peek() | It will return toppest element of stack. |

- 4) isEmpty() It is used to check stack is empty or not.
 5) search(Object o) It will return offset value if element is found otherwise it will return -1.

```

import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        Stack<String> s=new Stack<String>();
        s.push("A");
        s.push("B");
        s.push("C");
        System.out.println(s); // [A,B,C]

        s.pop();
        System.out.println(s); // [A,B]

        System.out.println(s.peek()); // B

        System.out.println(s.isEmpty()); // false

        System.out.println(s.search("Z")); // -1

        System.out.println(s.search("A")); // 2
    }
}
  
```

Set

It is a child interface of Collection interface.

If we want to represent group of individual objects in a single entity where duplicate objects are not allowed and order is not preserved then we need to use Set interface.

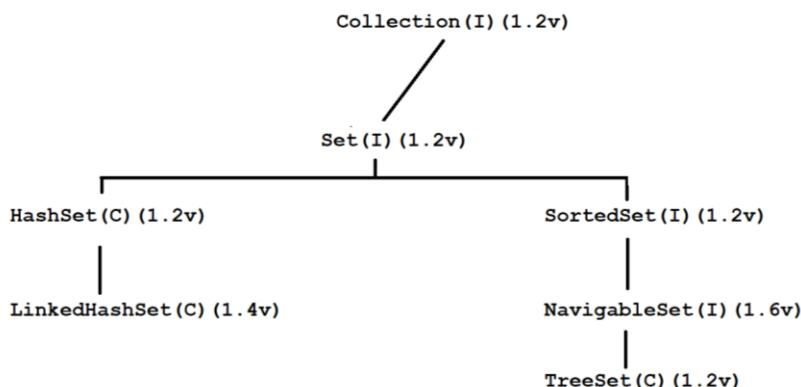


Diagram: java34.1

HashSet

The underlying data structure is Hashtable.

Duplicate objects are not allowed.

Order is not preserved because it will take hash code of an object.

Hetrogeneous objects are allowed.

Null insertion is possible.

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        HashSet hs=new HashSet();
        hs.add("one");
        hs.add("nine");
        hs.add("five");
        hs.add("eight");
        System.out.println(hs);//[nine, one, five, eight]
        hs.add("one");
        System.out.println(hs);//[nine,one,five,eight]
        hs.add(10);
        System.out.println(hs);//[nine, one, 10, five, eight]
        hs.add(null);
        System.out.println(hs);//[null, nine, one, 10, five, eight]
    }
}
```

ex:

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        HashSet<String> hs=new HashSet<String>();
        hs.add("one");
        hs.add("nine");
        hs.add("five");
        hs.add("eight");
        System.out.println(hs);//[nine, one, five, eight]
        hs.add("one");
        System.out.println(hs);//[nine,one,five,eight]
        hs.add(null);
        System.out.println(hs);//[null, nine, one, five, eight]
    }
}
```

LinkedHashSet

LinkedHashSet is a child class of HashSet class.

LinkedHashSet is exactly same as HashSet class with following differences.

HashSet

The underlying data structure is Hashtable.
Insertion order is not preserved.
It is introduced in 1.2v.

LinkedHashSet

The underlying data structure is Hashtable and LinkedList.
Insertion order is preserved.
It is introduced in 1.4v.

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        LinkedHashSet<String> lhs=new LinkedHashSet<String>();
        lhs.add("one");
        lhs.add("nine");
        lhs.add("five");
        lhs.add("eight");
        System.out.println(lhs);//[one,nine,five,eight]
        lhs.add("one");
        System.out.println(lhs);//[one,nine,five,eight]
        lhs.add(null);
        System.out.println(lhs);//[one,nine,five,eight,null]
    }
}
```

TreeSet

The underlying data structure is Balanced Tree.

Duplicate objects are not allowed.

Insertion order is not preserved because it will take sorting order of an object.

Heterogenous objects are not allowed if we insert then we will get ClassCastException.

For empty TreeSet if we trying to insert null then we will get NullPointerException.

After inserting the elements if we are trying to insert null then we will get NullPointerException.

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        TreeSet ts=new TreeSet();
        ts.add(10);
        ts.add(1);
        ts.add(5);
        ts.add(7);
        ts.add(3);
        System.out.println(ts);//[1,3,5,7,10]
        ts.add(10);
        System.out.println(ts);//[1,3,5,7,10]
        //ts.add("hi");
    }
}
```

```

        //System.out.println(ts); //R.E ClassCastException
        //ts.add(null);
        //System.out.println(ts); // R.E NullPointerException

    }
}

```

Q) What is the difference between Comparable and Comparator interface.

Comparable

Comparable interface present in java.lang package.

It contains only one method i.e compareTo() method.

ex: compareTo(Object obj1, Object obj2);
 It will return -ve if obj1 comes before obj2.
 It will return +ve if obj1 comes after obj2.
 it will return 0 if both objects are same.

If we depend upon default natural sorting then we need to use Comparable interface.

class Test

```

{
    public static void main(String[] args)
    {
        System.out.println("A".compareTo("Z")); // -25
        System.out.println("Z".compareTo("A")); // 25
        System.out.println("K".compareTo("K")); // 0
    }
}

```

Comparator

Comparator is an interface which is present in java.util package.

Comparator interface contains following two methods. i.e equals() and compare() method.

ex: public abstract int compare(Object obj1, Object obj2);

 It will return +ve if obj1 comes before obj2.
 It will return -ve if obj1 comes after obj2.
 It will return 0 if both objects are same.

 public abstract boolean equals(java.lang.Object);

Implementation of equals() method is optional because it is present in Object class so it is available through inheritance.

Implementation of compare() method is mandatory.

If we depends upon customized sorting order then we need to use Comparator interface.

import java.util.*;

class Test

```

{
    public static void main(String[] args)
    {
        TreeSet<Integer> ts=new TreeSet<Integer>(new MyComparator());
    }
}

```

```

        ts.add(10);
        ts.add(1);
        ts.add(5);
        ts.add(7);
        ts.add(3);
        System.out.println(ts);//10 7 5 3 1
    }
}
class MyComparator implements Comparator
{
    public int compare(Object obj1, Object obj2)
    {
        Integer i1=(Integer)obj1;
        Integer i2=(Integer)obj2;
        if(i1<i2)
            return 1;
        else if(i1>i2)
            return -1;
        else
            return 0;
    }
}

```

ex:

```

import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        TreeSet<Integer> ts=new TreeSet<Integer>(new MyComparator());
        ts.add(10);
        ts.add(1);
        ts.add(5);
        ts.add(7);
        ts.add(3);
        System.out.println(ts);//1 3 5 7 10
    }
}
class MyComparator implements Comparator
{
    public int compare(Object obj1, Object obj2)
    {
        Integer i1=(Integer)obj1;
        Integer i2=(Integer)obj2;
        if(i1<i2)
            return -1;
    }
}

```

```

        else if(i1>i2)
            return 1;
        else
            return 0;
    }
}

```

Map

It is not a child interface of Collection interface.

If we want to represent group of individual objects in key,value pair then we need to use Map interface.

Key can't be duplicated but value can be duplicated.

Here key and value both must be objects.

Each key and value pair is called one entry.

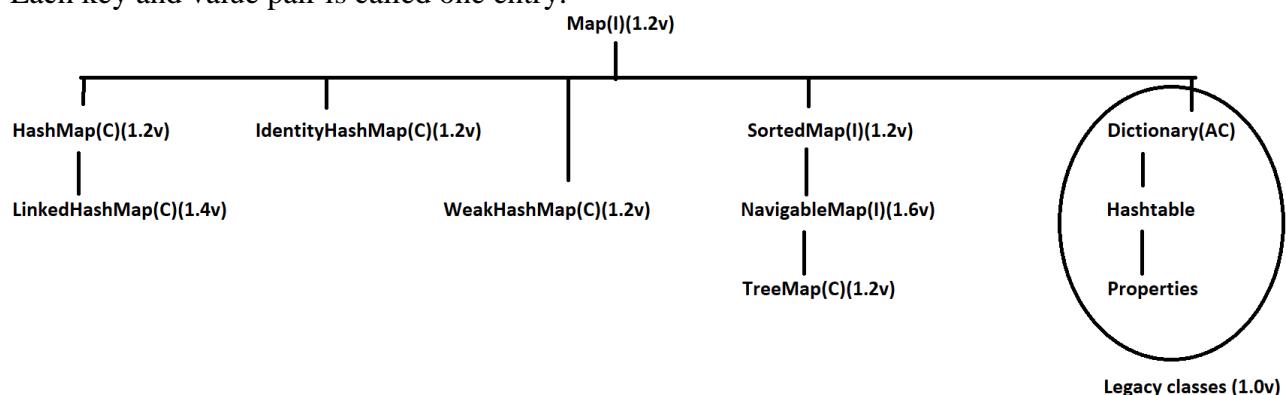


Diagram: java35.1

HashMap

The underlying data structure is Hashtable.

Key can't be duplicated but value can be duplicated.

Insertion order is not preserved because it will take hash code of the key.

Heterogeneous objects are allowed for both key and value.

Null insertion is possible for both key and value.

```

import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        HashMap<String,String> hm=new HashMap<String,String>();
        hm.put("one","raja");
        hm.put("nine","nelson");
        hm.put("six","jose");
        hm.put("eight","Erick");
        System.out.println(hm); // {nine=nelson, six=jose, one=raja, eight=Erick}
        hm.put("one","jojo");
        System.out.println(hm); // {nine=nelson, six=jose, one=jojo, eight=Erick}
        hm.put(null,null);
    }
}

```

```

        System.out.println(hm);//{null=null,      nine=nelson,      six=jose,      one=jojo,
eight=Erick}
    }
}

ex:
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        HashMap<String,String> hm=new HashMap<String,String>();
        hm.put("one","raja");
        hm.put("nine","nelson");
        hm.put("six","jose");
        hm.put("eight","Erick");

        Set s=hm.keySet();
        System.out.println(s);//[nine, six, one, eight]

        Collection c=hm.values();
        System.out.println(c);//[nelson, jose, raja, Erick]

        Set s1=hm.entrySet();
        System.out.println(s1);//[nine=nelson, six=jose, one=raja, eight=Erick]
    }
}

```

LinkedHashMap

It is a child class of `HashMap` class.

`LinkedHashMap` is exactly same as `HashMap` class with following differences.

<u>HashMap</u>	<u>LinkedHashMap</u>
The underlying data structure is Hashtable.	The underlying data structure is Hashtable and <code>LinkedList</code> .
Insertion order is not preserved.	Insertion order is preserved.
Introduced in 1.2v.	Introduced in 1.4v.

```

import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        LinkedHashMap<String,String> lhm=new LinkedHashMap<String,String>();
        lhm.put("one","raja");
        lhm.put("nine","nelson");
        lhm.put("six","jose");
    }
}

```

```

        lhm.put("eight","Erick");
        System.out.println(lhm); // {one=raja, nine=nelson, six=jose, eight=Erick}

    }
}

```

TreeMap

The underlying datastructure is RED BLACK TREE.

Key can't be duplicated but value can be duplicated.

Insertion order is not preserved because it will take sorting order of the key.

If we depend upon default natural sorting order then key can be homogeneous and comparable.

If we depend upon customized sorting order then key can be heterogeneous and non-comparable.

Key can't be null but value can be null.

```

import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        TreeMap<Integer,String> tm=new TreeMap<Integer,String>();
        tm.put(10,"ten");
        tm.put(1,"one");
        tm.put(5,"five");
        tm.put(7,"seven");
        tm.put(3,"three");
        System.out.println(tm); // {1=one, 3=three, 5=five, 7=seven, 10=ten}
        tm.put(4,null);
        System.out.println(tm); // {1=one, 3=three, 4=null, 5=five, 7=seven, 10=ten}
        //tm.put(null,"nine");
        //System.out.println(tm); // R.E NullPointerException
    }
}

```

Hashtable

The underlying datastructure is Hashtable.

Key can't be duplicated but value can be duplicated.

Insertion order is not preserved because it will take descending order of key.

Heterogeneous objects are allowed for both key and value.

Key and value can't be null.

```

import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        Hashtable<Integer,String> ht=new Hashtable<Integer,String>();
        ht.put(10,"ten");

```

```

ht.put(1,"one");
ht.put(5,"five");
ht.put(7,"seven");
ht.put(3,"three");
System.out.println(ht);//{10=ten, 7=seven, 5=five, 3=three, 1=one}
//ht.put(4,null);
//System.out.println(ht);//R.E NullPointerException
//ht.put(null,"four");
//System.out.println(ht);//R.E NullPointerException

}

}

```

Types of Cursors

Cursor is used to read objects one by one from Collections.

We have three types of cursors in Java.

- 1) Enumeration
- 2) Iterator
- 3) ListIterator

1)Enumeration

It is used to read objects one by one from Legacy Collection objects.

We can create Enumeration object as follow.

ex: Enumeration e=v.elements();

Enumeration interface contains following two methods.

ex: public boolean hasMoreElements();
 public Object nextElement();

import java.util.*;

class Test

{

```

    public static void main(String[] args)
    {
        Vector v=new Vector();
        for(int i=1;i<=10;i++)
        {
            v.add(i);
        }
        System.out.println(v);//[1,2,3,4,5,6,7,8,9,10]
    }
}
```

```

    Enumeration e=v.elements();
    while(e.hasMoreElements())
    {
        Integer i=(Integer)e.nextElement();
        System.out.println(i);
    }
}
```

Limitations with Enumeration

Using Enumeration we can read objects one by one only from legacy Collection objects.Hence it is not a universal cursor.

Using Enumeration interface we can perform read operation but not remove operation.

To overcome this limitation we need to use Iterator interface.

2)Iterator

It is used to read objects one by one from any Collection object.Hence it is a universal cursor.

Using Iterator interface we can perform read and remove operations.

We can create Iterator object as follow.

ex: Iterator itr=al.iterator();

Iterator interface contains following three methods.

ex: public boolean hasNext()

 public Object next()

 public void remove()

```
import java.util.*;
```

```
class Test
```

```
{
```

```
    public static void main(String[] args)
    {
```

```
        ArrayList al=new ArrayList();
        for(int i=1;i<=10;i++)
    {
```

```
            al.add(i);
    }
```

```
    System.out.println(al);//[1,2,3,4,5,6,7,8,9,10]
```

```
    Iterator itr=al.iterator();
    while(itr.hasNext())
    {
```

```
        Integer i=(Integer)itr.next();
        if(i%2==0)
```

```
        {
```

```
            itr.remove();
        }
```

```
}
```

```
    System.out.println(al);//[1,3,5,7,9]
```

```
}
```

```
}
```

Limitations with Iterator

> Using Enumeration and Iterator we can read objects only in forward direction but not in backward direction.Hence they are not bi-directional cursors.

> Using Iterator interface we can perform read and remove operation but not adding and replacement of new objects.

> To overcome this limitation we need to use ListIterator interface.

3)ListIterator

It is a child interface of Iterator interface.

ListIterator is used to read objects only from List Collection objects.Hence it is a bi-directional cursor.

Using ListIterator we can perform read, remove, adding and replacement of new objects.

We can create ListIterator object as follow.

ex: ListIterator litr=al.listIterator();

ListIterator interface contains following nine methods.

ex: public boolean hasNext()

public Object next()

public boolean hasPrevious()

public Object previous()

public void add(Object o)

public void set(Object o)

public int nextIndex()

public int previousIndex()

import java.util.*;

class Test

{

 public static void main(String[] args)

 {

 ArrayList al=new ArrayList();

 al.add("venki");

 al.add("chiru");

 al.add("bala");

 al.add("nag");

 System.out.println(al);//[venki,chiru,bala,nag]

 ListIterator litr=al.listIterator();

 while(litr.hasNext())

 {

 String s=(String)litr.next();

 System.out.println(s);

 }

}

ex:

import java.util.*;

class Test

{

 public static void main(String[] args)

 {

 ArrayList al=new ArrayList();

 al.add("venki");

```

al.add("chiru");
al.add("bala");
al.add("nag");
System.out.println(al);//[venki,chiru,bala,nag]

ListIterator litr=al.listIterator();
while(litr.hasNext())
{
    String s=(String)litr.next();
    if(s.equals("bala"))
    {
        litr.remove();
    }
}
System.out.println(al);//[venki,chiru,nag]

}

ex:
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        ArrayList al=new ArrayList();
        al.add("venki");
        al.add("chiru");
        al.add("bala");
        al.add("nag");
        System.out.println(al);//[venki,chiru,bala,nag]

        ListIterator litr=al.listIterator();
        while(litr.hasNext())
        {
            String s=(String)litr.next();
            if(s.equals("bala"))
            {
                litr.add("pavan");
            }
        }
        System.out.println(al);//[venki, chiru, bala, pavan, nag]
    }
}

```

```

ex:
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        ArrayList al=new ArrayList();
        al.add("venki");
        al.add("chiru");
        al.add("bala");
        al.add("nag");
        System.out.println(al);//[venki,chiru,bala,nag]

        ListIterator litr=al.listIterator();
        while(litr.hasNext())
        {
            String s=(String)litr.next();
            if(s.equals("bala"))
            {
                litr.set("Allu");
            }
        }
        System.out.println(al);//[venki, chiru, Allu, nag]
    }
}

```

Interview Questions

Q)Write a java program to find out repeating words in a given string?

input: this is is java java class

output: this=1 , is=2 , java=2 , class=1

```

import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        String str="this is is java java class";

        LinkedHashMap<String,Integer> lhm=new LinkedHashMap<String,Integer>();

        String[] sarr=str.split(" ");

        for(String s:sarr)
        {
            if(lhm.get(s)!=null)
            {

```

```

        lhm.put(s,lhm.get(s)+1);
    }
    else
    {
        lhm.put(s,1);
    }
}
System.out.println(lhm);
}
}

```

Q)Write a java program to find out repeating alphabets in a given string?

input: java

output: j=1,a=2,v=1

import java.util.*;

class Test

{

 public static void main(String[] args)

 {

 String str="java";

 LinkedHashMap<Character, Integer>

 lhm=new

 LinkedHashMap<Character, Integer>();

 char[] carr=str.toCharArray();

 for(char c:carr)

 {

 if(lhm.get(c)!=null)

 {

 lhm.put(c,lhm.get(c)+1);

 }

 else

 {

 lhm.put(c,1);

 }

 }

 System.out.println(lhm);

 }

}

Q)Write a java program to check given string is balanced or not?

input: {[()]}

output: It is balanced string

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        String str=" {[()]}";

        if(isBalanced(str))
            System.out.println("It is a balanced string");
        else
            System.out.println("It is not a balanced string");
    }
    //callie method
    public static boolean isBalanced(String str)
    {
        Stack<Character> stack=new Stack<Character>();

        char[] carr=str.toCharArray();

        for(char c:carr)
        {
            if(c=='{' || c=='[' || c=='(')
                stack.push(c);
            else if(c=='}' || !stack.isEmpty() || stack.peek()=='{')
                stack.pop();
            else if(c==']' || !stack.isEmpty() || stack.peek()=='[')
                stack.pop();
            else if(c==')' || !stack.isEmpty() || stack.peek()=='(')
                stack.pop();
            else
                return false;
        }

        return stack.isEmpty();
    }
}
```

Q) Types of Data structure in java?

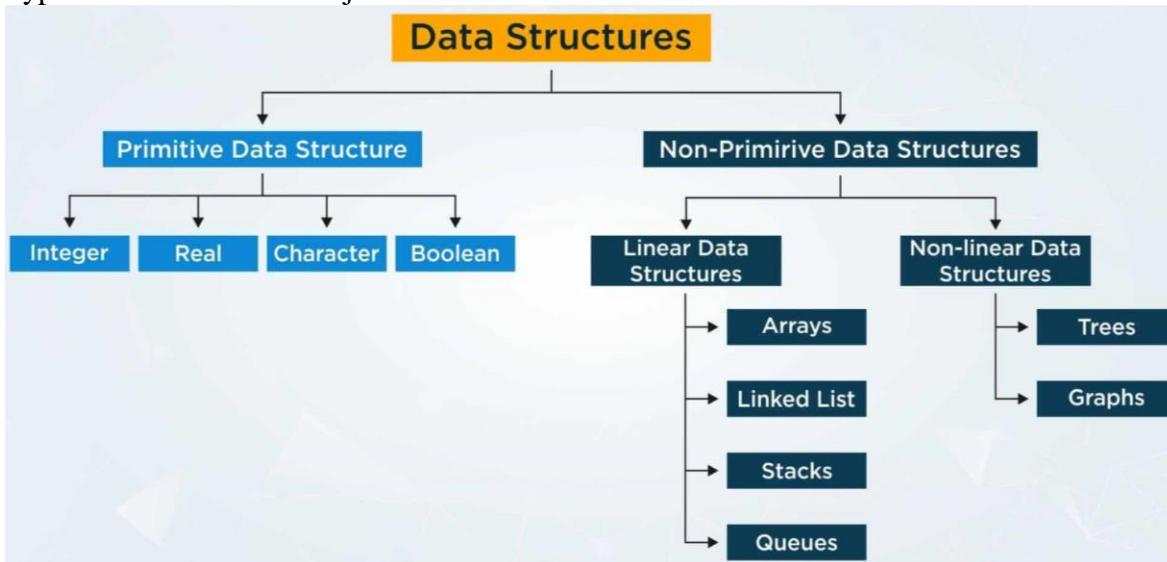


Diagram: java36.1

Q) Write a java program to display unique and duplicate elements from given array?

Input: 5 1 2 3 3 9 1 7 6 4 4 3

output: Unique : 5 1 2 3 9 7 6 4

Duplicates : 1 3 4

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        int[] arr={5,1,2,3,3,9,1,7,6,4,4,3};

        Set<Integer> duplicate=new LinkedHashSet<Integer>();
        Set<Integer> unique=new LinkedHashSet<Integer>();

        //for each loop
        for(int ele:arr)
        {
            if(!unique.add(ele))
            {
                duplicate.add(ele);
            }
            unique.add(ele);
        }

        System.out.println("Unique : "+unique);
        System.out.println("Duplicate :" +duplicate);
    }
}
```

Q)Write a java program to display unique elements from given array?

Input: 5 1 2 3 3 9 1 7 6 4 4 3

output: 5 1 2 3 9 7 6 4

```
import java.util.*;
```

```
class Test
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        int[] arr={5,1,2,3,3,9,1,7,6,4,4,3};
```

```
        Set<Integer> unique=new LinkedHashSet<Integer>();
```

```
        //for each loop
```

```
        for(int ele:arr)
```

```
        {
```

```
            unique.add(ele); // 5 1 2 3 9 7 6 4
```

```
        }
```

```
        //converting collection to array
```

```
        int[] resArr=new int[unique.size()];
```

```
        int j=0;
```

```
        for(Integer i:unique)
```

```
        {
```

```
            resArr[j++]=i;
```

```
        }
```

```
        //display
```

```
        for(int ele:resArr)
```

```
        {
```

```
            System.out.print(ele+" ");
```

```
        }
```

```
}
```

Multithreading

Q)What is the difference between Thread and Process?

Thread

A thread is a light weight sub-process.

We can run multiple threads concurrently.

One thread can communicate with another thread.

ex: In java a class is one thread

In java a constructor is one thread

In java a block is one thread

Process

A process is a collection of threads.

We can run multiple process concurrently.

One process can't communicate with another process because it is independent.

ex: Taking a java class by using zoom meeting is one process

Downloading a file from internet is one process

typing the notes in editor is one process

Multitasking

Executing several task simultaneously such concept is called multitasking.

We have two types of multitasking.

1)Process based multitasking

Executing several task simultaneously where each task is an independent process such type of multitasking is called process based multitasking.

2)Thread based multitasking

Executing several task simultaneously where each task is a same part of a program such type of multitasking is called thread based multitasking.

Multithreading

Executing several threads simultaneously such concept is called multithreading.

In multithreading only 10% of work should be done by a programmer and 90% of work will be done by a JAVA API.

The main important application area of multithreading are.

1) To implement multimedia graphics.

2) To develop video games

3) To develop animations.

Ways to create a thread in java

There are two ways to create or start or instantiate a thread in java.

1) By extending Thread class

2) By implementing Runnable interface

1) By extending Thread class

```
class MyThread extends Thread
{
    public void run()
    {
        for(int i=1;i<=5;i++)
        {
            System.out.println("Child-Thread");
        }
    }
}
class Test
{
    public static void main(String[] args)
    {
        //instantiate a thread
        MyThread t=new MyThread();
```

```

//start a thread
t.start();

for(int i=1;i<=5;i++)
{
    System.out.println("Parent-Thread");
}
}
}

```

case1: Thread Scheduler

If multiple threads are waiting for execution which thread will execute will decide by thread scheduler.

What algorithm, behaviour and mechanism used by thread scheduler is depends upon JVM vendor.

Hence we can't expect any execution order or exact output in multithreading.

case2: Difference between t.start() and t.run()

If we invoke t.start() method then a new object will created which is responsible to execute run() method automatically.

class MyThread extends Thread

```

{
    public void run()
    {
        for(int i=1;i<=5;i++)
        {
            System.out.println("Child-Thread");
        }
    }
}
```

class Test

```

{
    public static void main(String[] args)
    {
        //instantitate a thread
        MyThread t=new MyThread();

        //start a thread
        t.start();

        for(int i=1;i<=5;i++)
        {
            System.out.println("Parent-Thread");
        }
    }
}
```

```
}
```

If we invoke t.run() method then no new thread will be created but run() method will execute just like normal method.

```
class MyThread extends Thread
{
    public void run()
    {
        for(int i=1;i<=5;i++)
        {
            System.out.println("Child-Thread");
        }
    }
}
class Test
{
    public static void main(String[] args)
    {
        //instantitate a thread
        MyThread t=new MyThread();

        //no new thread
        t.run();

        for(int i=1;i<=5;i++)
        {
            System.out.println("Parent-Thread");
        }
    }
}
```

case3: If we won't override run() method

If we won't override run() method then t.start() method will execute Thread class run() method automatically.

Thread class run() method is a empty implementation.

```
class MyThread extends Thread
{
}

class Test
{
    public static void main(String[] args)
    {
        //instantitate a thread
        MyThread t=new MyThread();
```

```

//new thread
t.start();

for(int i=1;i<=5;i++)
{
    System.out.println("Parent-Thread");
}
}
}

```

case4: If we overload run() method

If we overload run() method then t.start() method always execute run() method with 0-argument method.

class MyThread extends Thread

```

{
    public void run()
    {
        System.out.println("0-arg method");
    }
    public void run(int i)
    {
        System.out.println("int-arg method");
    }
}

```

class Test

```

{
    public static void main(String[] args)
    {
        //instantitate a thread
        MyThread t=new MyThread();

        //new thread
        t.start();

        for(int i=1;i<=5;i++)
        {
            System.out.println("Parent-Thread");
        }
    }
}

```

case5: Life cycle of a thread

Once if we create a thread object then our thread will be in new or born state.

Once if we call t.start() method then our thread goes to ready/runnable state.

If Thread scheduler allocates to CPU then out thread enters to running state.

Once the run() method execution is completed then our thread goes to dead state.

```

MyThread t=new MyThread(); //instantiate a thread
t.start();

```

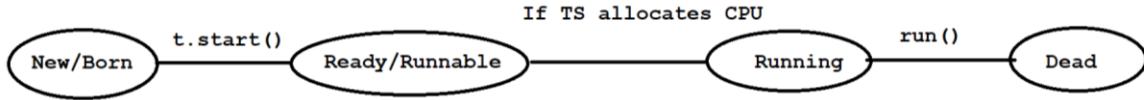


Diagram: java36.2

2)By implementing Runnable interface

```

class MyRunnable implements Runnable
{
    public void run()
    {
        for(int i=1;i<=5;i++)
        {
            System.out.println("Child-Thread");
        }
    }
}
class Test
{
    public static void main(String[] args)
    {
        MyRunnable r=new MyRunnable();

        Thread t=new Thread(r); // r is a targatable interface

        t.start();

        for(int i=1;i<=5;i++)
        {
            System.out.println("Parent-Thread");
        }
    }
}

```

Setting and Getting name of a thread

In java , every thread has a name explicitly provided by the programmer and automatically generated by JVM.

We have following methods to set and get name of a thread.

ex: public final void setName(String name);
 public final String getName();

class MyThread extends Thread

```

{
}
```

```

class Test
{
    public static void main(String[] args)
    {
        System.out.println(Thread.currentThread().getName());//main

        MyThread t=new MyThread();
        System.out.println(t.getName());//Thread-0

        Thread.currentThread().setName("Parent-Thread");
        System.out.println(Thread.currentThread().getName());//Parent-Thread

        t.setName("Child-Thread");
        System.out.println(t.getName());//Child-Thread
    }
}

```

Thread priority

In java, every thread has a priority explicitly provided by the programmer or automatically generated by JVM.

The valid range of thread priority is 1 to 10. Where 1 is a least priority and 10 is a highest priority.

Thread class defines following constants as a thread priority.

ex: Thread.MAX_PRIORITY - 10
 Thread.MIN_PRIORITY - 1
 Thread.NORM_PRIORITY - 5

We don't have such constants like LOW_PRIORITY and HIGH_PRIORITY.

A thread which is having highest priority will be executed first.

Thread scheduler uses thread priority while allocating to CPU.

If multiple threads having same priority then we can't expect any execution order.

If we take more than 10 priority then we will get IllegalArgumentException.

We have following methods to set and get thread priority.

ex: public final void setPriority(int priority);
 public final int getPriority();

class MyThread extends Thread

{
 }

class Test

{

 public static void main(String[] args)
 {
 System.out.println(Thread.currentThread().getPriority());//5

 MyThread t=new MyThread();
 System.out.println(t.getPriority());//5

```

        Thread.currentThread().setPriority(9);
        System.out.println(Thread.currentThread().getPriority());//9

        t.setPriority(4);
        System.out.println(t.getPriority());//4
    }
}

```

Various Ways to prevent a thread in java

In three ways we can prevent(stop) a thread in java.

- 1)yield()
- 2)join()
- 3)sleep()

1)yield()

It will pause the current execution thread and gives the chance to other thread who is having same priority.

If there is no waiting thread or low priority thread then same thread will continue it's execution.

If multiple threads having same priority then we can't expect any execution order or exact output. The thread which is yielded , when it will chance for execution is depends upon mercy of thread scheduler.

ex: public static native void yield();

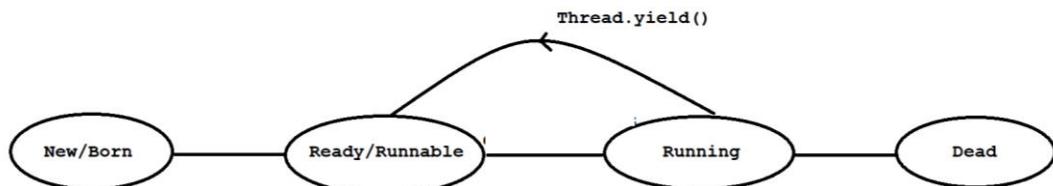


Diagram: java37.1

```

class MyThread extends Thread
{
    public void run()
    {
        for(int i=1;i<=5;i++)
        {
            System.out.println("Child-Thread");
        }
    }
}

class Test
{
    public static void main(String[] args)
    {
        MyThread t=new MyThread();
        t.start();
        for(int i=1;i<=5;i++)
        {
    }
}

```

```

        Thread.currentThread().yield();
        System.out.println("Parent-Thread");
    }
}
}

```

2)join()

If a thread wants to wait until the completion of some other threads then we need to use join() method.

A join() method will throw one checked exception so we must and should handle that exception by using try and catch block or by using throws statement.

ex: public final void join()throws InterruptedException
 public final void join(long ms)throws InterruptedException
 public final void join(long ms,int ns)throws InterruptedException

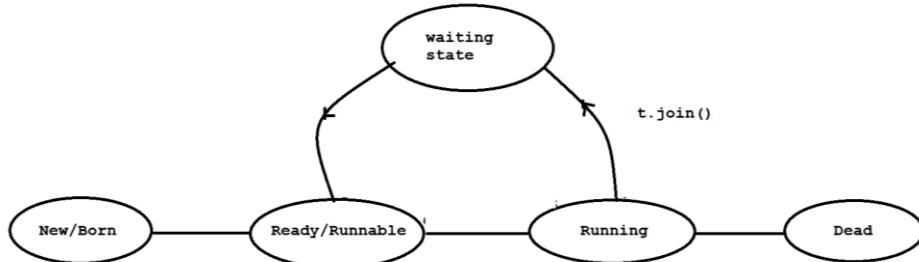


Diagram:java37.2

```

class MyThread extends Thread
{
    public void run()
    {
        for(int i=1;i<=5;i++)
        {
            System.out.println("Child-Thread");
        }
    }
}
class Test
{
    public static void main(String[] args)throws InterruptedException
    {
        MyThread t=new MyThread();
        t.start();
        t.join();
        for(int i=1;i<=5;i++)
        {
            System.out.println("Parent-Thread");
        }
    }
}

```

3)sleep()

If a thread don't want to perform any operation on particular amount of time then we need to use sleep() method.

A sleep() will throw one checked exception so we must and should handle that exception by using try and catch block or by using throws statement.

ex:

```
public static native void sleep()throws InterruptedException  
public static native void sleep(long ms)throws InterruptedException  
public static native void sleep(long ms,int ns)throws InterruptedException
```

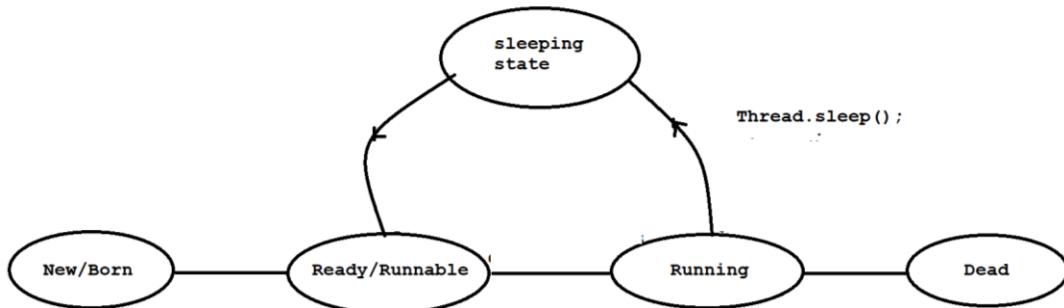


Diagram:java37.3

```
class MyThread extends Thread  
{  
    public void run()  
    {  
        for(int i=1;i<=5;i++)  
        {  
            System.out.println("Child-Thread");  
            try  
            {  
                Thread.sleep(2000);  
            }  
            catch (InterruptedException ie)  
            {  
                ie.printStackTrace();  
            }  
        }  
    }  
}  
class Test  
{  
    public static void main(String[] args)  
    {  
        MyThread t=new MyThread();  
        t.start();  
        for(int i=1;i<=5;i++)  
        { System.out.println("Parent-Thread"); }  
    }  
}
```

Daemon Thread

It is a service provider thread which provides services to user threads.

The life of daemon thread is completely depends upon user threads. When user thread died then daemon thread will terminate automatically.

There are many daemon thread are running internally like Garbage Collector, Finalizer and etc.

To start a daemon thread we need to use setDaemon(true) method.

To check either thread is a daemon or not we need to use isDaemon() method.

```
class MyThread extends Thread
```

```
{
```

```
    public void run()
```

```
{
```

```
    for(int i=1;i<=5;i++)
```

```
{
```

```
        System.out.println(Thread.currentThread().isDaemon());
```

```
        System.out.println("Child-Thread");
```

```
}
```

```
}
```

```
}
```

```
class Test
```

```
{
```

```
    public static void main(String[] args)
```

```
{
```

```
        MyThread t=new MyThread();
```

```
        t.setDaemon(true);
```

```
        t.start();
```

```
        for(int i=1;i<=5;i++)
```

```
{
```

```
            System.out.println("Parent-Thread");
```

```
}
```

```
}
```

```
}
```

Problem without synchronization

If there is a no synchronization then we will face following problems.

1) Data inconsistency.

2) Thread interference.

```
class Table
```

```
{
```

```
    void printTable(int n)
```

```
{
```

```
    for(int i=1;i<=5;i++)
```

```
{
```

```
        System.out.println(n*i);
```

```
        try
```

```
{
```

```
            Thread.sleep(2000);
```

```

        }
        catch (InterruptedException ie)
        {
            ie.printStackTrace();
        }
    }
}

class MyThread1 extends Thread
{
    Table t;

    MyThread1(Table t)
    {
        this.t=t;
    }

    public void run()
    {
        t.printTable(5);
    }
}

class MyThread2 extends Thread
{
    Table t;

    MyThread2(Table t)
    {
        this.t=t;
    }

    public void run()
    {
        t.printTable(10);
    }
}

class Test
{
    public static void main(String[] args)
    {
        Table obj=new Table();
        MyThread1 t1=new MyThread1(obj);
        MyThread2 t2=new MyThread2(obj);
        t1.start();
        t2.start();
    }
}

```

synchronization

A synchronized keyword is applicable for methods and blocks.

A synchronization is allowed one thread to execute given object.Hence we achieve thread safety.

The main advantage of synchronization is we solve data inconsistency problem.

The main disadvantage of synchronization is ,it will increase waiting time of a thread which reduce the performance of the system.

If there is no specific requirement then it is never recommended to use synchronization concept.
synchronization internally uses lock mechanism.

Whenever a thread wants to access object , first it has to acquire lock of an object and thread will release the lock when it completes it's task.

When a thread wants to execute synchronized method.It automatically gets the lock of an object.

When one thread is executing synchronized method then other threads are not allowed to execute other synchronized methods in a same object concurrently.But other threads are allowed to execute non-synchronized method concurrently.

```
class Table
```

```
{
```

```
    synchronized void printTable(int n)
```

```
{
```

```
    for(int i=1;i<=5;i++)
```

```
{
```

```
        System.out.println(n*i);
```

```
        try
```

```
{
```

```
            Thread.sleep(2000);
```

```
}
```

```
        catch (InterruptedException ie)
```

```
{
```

```
            ie.printStackTrace();
```

```
}
```

```
}
```

```
}
```

```
class MyThread1 extends Thread
```

```
{
```

```
    Table t;
```

```
    MyThread1(Table t)
```

```
{
```

```
        this.t=t;
```

```
}
```

```
    public void run()
```

```
{
```

```
        t.printTable(5);
```

```
}
```

```
}
```

```
class MyThread2 extends Thread
```

```

{
    Table t;
    MyThread2(Table t)
    {
        this.t=t;
    }
    public void run()
    {
        t.printTable(10);
    }
}

class Test
{
    public static void main(String[] args)
    {
        Table obj=new Table();
        MyThread1 t1=new MyThread1(obj);
        MyThread2 t2=new MyThread2(obj);

        t1.start();
        t2.start();
    }
}

```

synchronized block

If we want to perform synchronization on specific resource of a program then we need to use synchronization.

ex: If we have 100 lines of code and if we want to perform synchronization only for 10 lines then we need to use synchronized block.

If we keep all the logic in synchronized block then it will act as a synchronized method.

class Table

```

{
    void printTable(int n)
    {
        synchronized(this)
        {
            for(int i=1;i<=5;i++)
            {
                System.out.println(n*i);
                try
                {
                    Thread.sleep(2000);
                }
                catch (InterruptedException ie)
                {

```

```
        ie.printStackTrace();
    }
}
}//sync
}

class MyThread1 extends Thread
{
    Table t;
    MyThread1(Table t)
    {
        this.t=t;
    }
    public void run()
    {
        t.printTable(5);
    }
}

class MyThread2 extends Thread
{
    Table t;
    MyThread2(Table t)
    {
        this.t=t;
    }
    public void run()
    {
        t.printTable(10);
    }
}

class Test
{
    public static void main(String[] args)
    {
        Table obj=new Table();
        MyThread1 t1=new MyThread1(obj);
        MyThread2 t2=new MyThread2(obj);

        t1.start();
        t2.start();
    }
}
```

3)Static synchronization

In static synchronization the lock will be on class but not on object.

If we declare any static method as synchronized then it is called static synchronization method.

class Table

{

 static synchronized void printTable(int n)

{

 for(int i=1;i<=5;i++)

{

 System.out.println(n*i);

 try

{

 Thread.sleep(2000);

}

 catch (InterruptedException ie)

{

 ie.printStackTrace();

}

}

}

class MyThread1 extends Thread

{

 public void run()

{

 Table.printTable(5);

}

}

class MyThread2 extends Thread

{

 public void run()

{

 Table.printTable(10);

}

}

class Test

{

 public static void main(String[] args)

{

```

        MyThread1 t1=new MyThread1();
        MyThread2 t2=new MyThread2();

        t1.start();
        t2.start();
    }
}

```

.Inter-Thread Communication

Two threads can communicate with one another by using wait(),notify() and notifyAll() method. The Thread which is expecting updations it has to wait() method and the thread which is performing updations it has to call notify() method.

wait(),notify() and notifyAll() method present in Object class but not in Thread class.

To call wait(),notify() and notifyAll() method our current thread must be in a synchronized area otherwise we will get IllegalMonitorStateException.

Once a thread calls wait() method on a given object ,1st it will release the lock of that object immediately and entered into waiting state.

Once a thread calls notify() and notifyAll() method on a given object.It will release the lock of that object but not immediately.

Except wait(),notify() and notifyAll() method ,there is no such concept where lock release can happen.

class MyThread extends Thread

```
{

```

```
    int total=0;

```

```
    public void run()
    {

```

```
        synchronized(this)
    {

```

```
            System.out.println("Child Thread started calculation");

```

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

```

```

            {

```

```
                total=total+i;

```

```

            }

```

```
            System.out.println("Child thread giving notification");

```

```
            this.notify();

```

```

        }
    }
}

```

class Test

```
{

```

```
    public static void main(String[] args) throws InterruptedException
    {

```

```
        MyThread t=new MyThread();

```

```
        t.start();

```

```
        synchronized(t)

```

```

    {
        System.out.println("Main Thread waiting for updating");
        t.wait();
        System.out.println("Main -Thread got notification ");
        System.out.println(t.total);
    }
}
}

```

DeadLock in java

DeadLock will occur in a situation when one thread is waiting to access object lock which is acquired by another thread and that thread is waiting to access object lock which is acquired by first thread.

Here both the threads are waiting release the thread but no body will release such situation is called DeadLock.

```

class Test
{
    public static void main(String[] args)
    {
        final String res1="hi";
        final String res2="bye";

        Thread t1=new Thread()
        {
            public void run()
            {
                synchronized(res1)
                {
                    System.out.println("Thread1: Locking Resource 1");
                    synchronized(res2)
                    {
                        System.out.println("Thread1: Locking Resource2");
                    }
                }
            }
        };
        Thread t2=new Thread()
        {
            public void run()
            {
                synchronized(res2)
                {
                    System.out.println("Thread2: Locking Resource 2");
                    synchronized(res1)
                    {

```

```

        System.out.println("Thread1: Locking Resource 1");
    }
}
};

t1.start();
t2.start();
}

}

```

Drawbacks of multithreading

- 1)DeadLock
- 2)Thread Starvation

Java 8 Features

- 1) java.time package
- 2) Functional interface
- 3) Lamda Expression
- 4) default methods in interface
- 5) static methods in interface
- 6) Stream API
- 7) forEach() method
- and etc.

Functional interface

Interface which contains only one abstract method is called functional interface.

It can have any number of default methods and static methods.

ex: Runnable ---- run()
 Comparable ---- compareTo()
 ActionListener --- actionPerformed()
 and etc.

Functional interface is also known as SAM or Single Abstract Method interface.

Functional interface is used to achieve functional programming.

ex: i=f1(){ }
 f1(f2(){ })
 { }

@FunctionalInterface annotation is used to declare function interface and it is optional.

syntax: @FunctionalInterface
 interface <interface_name>
 {
 - // 1 abstract method
 - // default methods
 - // static methods
 }

```

ex:
@interface A
{
    public abstract void m1();
}
class B implements A
{
    public void m1()
    {
        System.out.println("M1-Method");
    }
}
class Test
{
    public static void main(String[] args)
    {
        A a=new B();
        a.m1();
    }
}

```

```

ex:
@interface A
{
    public abstract void m1();
}
class Test
{
    public static void main(String[] args)
    {
        A a=new A()
        {
            public void m1()
            {
                System.out.println("From M1-Method");
            }
        };
        a.m1();
    }
}

```

Lamda Expression

Lamda expression introduced in java 8.
Lamda expression is used to concise the code.

Lamda expression we can use when we have functional interface.

Lamda expression consider as method not a class.

The main objective of lamda expression is to achieve functional programming.

Lamda expression does not allow name, returntype and modifier.

ex: java method

```
public void m1()
{
    System.out.println("Hello World");
}
```

lamda expression

```
()->
{
    System.out.println("Hello World");
};
```

ex:

```
@FunctionalInterface
interface A
{
    public abstract void m1();
}
```

```
class Test
{
    public static void main(String[] args)
    {
        A a=()->
        {
            System.out.println("M1-Method");
        };
        a.m1();
    }
}
```

ex:

```
@FunctionalInterface
interface A
{
    public abstract void m1(int i,int j);
}
class Test
{
    public static void main(String[] args)
    {
        A a=(int i,int j)->
        {
```

```

        System.out.println(i+j);
    };
    a.m1(10,20);
}
}

```

ex:

```

@FunctionalInterface
interface A
{
    public abstract String m1();
}
class Test
{
    public static void main(String[] args)
    {
        A a=()->
        {
            return "Hello World";
        };
        System.out.println(a.m1());
    }
}

```

default methods in interface

A default method introduced in java 8.

To declare default methods in interface we will use "default" keyword.

A default method is a non-abstract method.

A default method can be override.

ex:

```

@FunctionalInterface
interface A
{
    //abstract method
    public abstract void m1();

    //default method
    default void m2()
    {
        System.out.println("M2-Method");
    }
}
class B implements A
{
    public void m1()
    {

```

```
        System.out.println("M1-Method");
    }
}
class Test
{
    public static void main(String[] args)
    {
        A a=new B();
        a.m1();
        a.m2();
    }
}
```

ex:

```
@FunctionalInterface
interface A
{
    //abstract method
    public abstract void m1();

    //default method
    default void m2()
    {
        System.out.println("M2-Method");
    }
}
class B implements A
{
    public void m1()
    {
        System.out.println("M1-Method");
    }
    public void m2()
    {
        System.out.println("Override-M2-Method");
    }
}
class Test
{
    public static void main(String[] args)
    {
        A a=new B();
        a.m1();
        a.m2();
    }
}
```

With the help of default methods we can achieve multiple inheritance.

ex:

```
interface Right
{
    default void m1()
    {
        System.out.println("Right-M1-Method");
    }
}

interface Left
{
    default void m1()
    {
        System.out.println("Left-M1-Method");
    }
}

class Middle implements Right,Left
{
    public void m1()
    {
        System.out.println("Middle-M1-Method");
    }
}

class Test
{
    public static void main(String[] args)
    {
        Middle m=new Middle();
        m.m1();
    }
}

o/p: Middle-M1-Method
```

ex:

```
interface Right
{
    default void m1()
    {
        System.out.println("Right-M1-Method");
    }
}

interface Left
{
    default void m1()
    {
        System.out.println("Left-M1-Method");
    }
}
```

```

        }
    }
class Middle implements Right,Left
{
    public void m1()
    {
        Right.super.m1();
    }
}
class Test
{
    public static void main(String[] args)
    {
        Middle m=new Middle();
        m.m1();
    }
}
o/p:  Right-M1-Method

```

ex:

```

interface Right
{
    default void m1()
    {
        System.out.println("Right-M1-Method");
    }
}
interface Left
{
    default void m1()
    {
        System.out.println("Left-M1-Method");
    }
}
class Middle implements Right,Left
{
    public void m1()
    {
        Left.super.m1();
    }
}
class Test
{
    public static void main(String[] args)
    {
        Middle m=new Middle();

```

```
        m.m1();
    }
}

o/p: Left-M1-Method
```

static methods in interface

A static method introduced in java 8.

To declare static methods in interface we will use "static" keyword.

A static method is a non-abstract method.

A static method can't be override.

ex:

```
interface A
{
    //static method
    static void m1()
    {
        System.out.println("static-method");
    }
}

class Test
{
    public static void main(String[] args)
    {
        A.m1();
    }
}
```

Stream API

Stream API introduced in java 8.

A Stream is an interface which is present in `java.util.stream` package.

It is used to perform bulk operations on Collections.

```
import java.util.*;
import java.util.stream.*;
class Test
{
    public static void main(String[] args)
    {
        List<Integer> list=Arrays.asList(2,5,7,4,6,9,1);

        List<Integer> even=list.stream().filter(i->i%2==0).collect(Collectors.toList());

        System.out.println(even);
    }
}
```

```

ex:
import java.util.*;
import java.util.stream.*;
class Test
{
    public static void main(String[] args)
    {
        List<Integer> list=Arrays.asList(2,5,7,4,6,9,1);

        List<Integer> odd=list.stream().filter(i->i%2!=0).collect(Collectors.toList());

        System.out.println(odd);
    }
}

ex:
import java.util.*;
import java.util.stream.*;
class Test
{
    public static void main(String[] args)
    {
        List<Integer> list=Arrays.asList(2,5,7,4,6,9,1);

        List<Integer> newList=list.stream().sorted().collect(Collectors.toList());

        System.out.println(newList);
    }
}

ex:
import java.util.*;
import java.util.stream.*;
class Test
{
    public static void main(String[] args)
    {
        List<Integer> list=Arrays.asList(2,5,7,4,6,9,1);

        List<Integer>
newList=list.stream().sorted(Comparator.reverseOrder()).collect(Collectors.toList());

        System.out.println(newList);
    }
}

```

ex:

```
import java.util.*;
import java.util.stream.*;
class Test
{
    public static void main(String[] args)
    {
        List<Integer> list=Arrays.asList(2,5,7,4,6,9,1);

        long max=list.stream().max((i1,i2)->i1.compareTo(i2)).get();

        System.out.println(max);
    }
}
```

ex:

```
import java.util.*;
import java.util.stream.*;
class Test
{
    public static void main(String[] args)
    {
        List<Integer> list=Arrays.asList(2,5,7,4,6,9,1);

        long min=list.stream().min((i1,i2)->i1.compareTo(i2)).get();

        System.out.println(min);
    }
}
```

ex:

```
import java.util.*;
import java.util.stream.*;
class Test
{
    public static void main(String[] args)
    {
        List<Integer> marks=Arrays.asList(24,58,72,49,16,90,81);

        List<Integer> list=marks.stream().map(i->i+10).collect(Collectors.toList());

        System.out.println(list);
    }
}
```

```

ex:
import java.util.*;
import java.util.stream.*;
class Test
{
    public static void main(String[] args)
    {
        List<Integer> marks=Arrays.asList(24,58,72,49,16,90,81);

        long failed=marks.stream().filter(i->i<35).count();

        System.out.println(failed);
    }
}

```

forEach() method

The forEach() method in Java is a utility method to iterate over a Collection (list, set or map) or Stream.

```

import java.util.*;
import java.util.stream.*;
class Test
{
    public static void main(String[] args)
    {
        List<Integer> list=Arrays.asList(24,58,72,49,16,90,81);

        list.forEach(System.out::println);
    }
}

```

Interview Question

Q) Write a java program to sort employees by id using java 8 stream api?

```

import java.util.*;
import java.util.stream.*;

class Employee
{
    private int empId;
    private String empName;
    private double empSal;

    //default constructor
    Employee()
    {

    }
}

```

```
//parameterized constructor
Employee(int empId,String empName,double empSal)
{
    this.empId=empId;
    this.empName=empName;
    this.empSal=empSal;
}

//setter methods
public void setEmpId(int empId)
{
    this.empId=empId;
}
public void setEmpName(String empName)
{
    this.empName=empName;
}
public void setEmpSal(double empSal)
{
    this.empSal=empSal;
}

//getter methods
public int getEmpId()
{
    return empId;
}
public String getEmpName()
{
    return empName;
}
public double getEmpSal()
{
    return empSal;
}

}

class Test
{
    public static void main(String[] args)
    {
        List<Employee> employees=new ArrayList<Employee>();
        employees.add(new Employee(104,"Alan",4000d));
        employees.add(new Employee(101,"Jose",1000d));
        employees.add(new Employee(103,"Kelvin",3000d));
        employees.add(new Employee(102,"Nelson",2000d));
    }
}
```

```

List<Employee>
list=employees.stream().sorted(Comparator.comparingInt(Employee::getEmpId)).collect(Collectors.toList());

        list.forEach(employee    ->    System.out.println(employee.getEmpId()      +""
"+employee.getEmpName() +" "+employee.getEmpSal()));

    }
}

```

SDLC

SDLC stands for Software Development Life Cycle.

It is a process adopted by IT industry to develop accurate and quality of softwares.

There are six phases present in SDLC.

1. Feasibility study
2. Analysis
3. Designing
4. Coding
5. Testing
6. Delivery and Maintenance

1)Feasibility study

Feasibility study completed depends upon TELOS formula.

ex: T - Technical Feasibility
 E - Economic Feasibility
 L - Legal Feasibility
 O - Operational Feasibility
 S - Scheduled Feasibility

All the above information they will keep in a document called BDD.

BDD stands for Business Designed Document.

2)Analysis

In this phase , system analyst or product owner will be involved.

They will separate system requirements and software requirements.

They will keep this information in a document called SRS.

SRS stands for Software Requirement Specification.

3)Designing

Designing can be performed in two ways.

i)High level designing

Manager is responsible to perform high level designing.

In high level designing ,we will design main modules.

ii)Low level designing

Team Lead/Project Lead is responsible to perform low level designing.

In low level designing , we will design sub-module/child modules.

All this above information we will keep in a document called PDD/TDD.
Here PDD stands for Project Design Document.
Here TDD stands for Technical Design Document.

4)Coding

In coding phase, developers will involved.
Developers are responsible to generate the build(source code).
Once our build is ready then developer even responsible to perform white box testing.

5)Testing Phase

In this phase, testing team or QA team will involved.
The will test the code by using one software component called STLC.
STLC stands for Software Testing Life Cycle.
Testing Team or QA team is responsible to perform black box testing.

6)Delivery & maintainence phase

Before delivery we will perform UAT testing.
UAT stands for User Acceptance Testing.
UAT testing is classified into two types.
i)Alpha testing
ii)Beta testing

Project

Project is a collection of modules.
ex: customer module
login module
registration module
payment module
report generation module
and etc.

Every project contains two domains.

1. Technical Domain

Using which technology we developed our project.
ex: Java

2. Functional Domain

It describes state of a project.
ex: Healthcare domain
Insurance domain
Banking domain
ERP domain
and etc.

ORACLE

Types of data

We have two types of data.

1) Unstructured Data

Data which is not in readable format is called unstructured data.

In general, meaning less data is called unstructured data.

ex: 302 Lakemba SYD NSW AUS

2) Structured Data

Data which is in readable format is called structured data.

In general, meaning full data is called structured data.

ex: unit locality city state country
302 Lakemba SYD NSW AUS

Oracle

Oracle is one of the database which is used to store structured data.

Oracle is a product of Oracle Corporation.

Oracle is a case insensitive language.

Oracle is classified into two types.

Oracle

|

SQL

(Structured Query Language)

Develop by IBM

PL/SQL

(Procedural/Structured Query Language)

Develop by Oracle Corporation

Client-Server Architecture

In this architecture we will see how our data will store from frontend to backend.

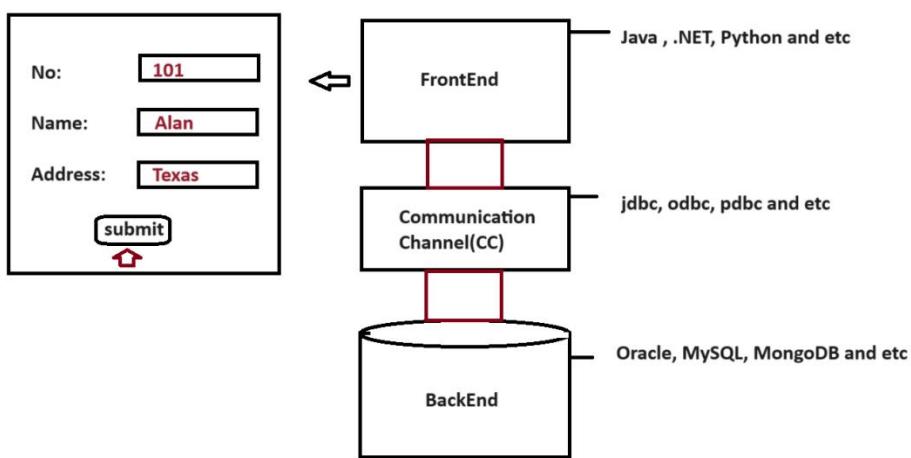


Diagram: oracle1.1

FrontEnd

The one which is visible to the enduser to perform some operations is called frontend.

ex: Java, .net, Python, Ruby, Perl and etc.

Communication Channel

It acts like a bridge between frontend and backend.

ex: JDBC - Java Database Connectivity

ODBC - Open Database Connectivity

PDBC - Python Database Connectivity

and etc

BackEnd

The one which is not visible to the enduser but it performs operations based on the instructions given by frontend is called backend.

ex: Oracle, MySQL, SQL Server, MongoDB, NoSQL, TeraData, DB2, Sybase and etc.

Management System

Management system is a software which is used to manage the database.

Management system will perform following activities very easily.

- 1) Inserting new data
- 2) Modifying existing data
- 3) Deleting unnecessary data
- 4) Selecting required data

DBMS

A database along with software which is used to manage the database is called database management system.

RDBMS

Any database which is developed based on relational theories which is used to manage the database is called relational database management system.

ex: Oracle, MySQL, SQL Server, PostgreSQL and etc.

SQL

SQL stands for Structured Query Language which is pronounce as SEQUEL.

This language is used to communicate with oracle database.

It is a command based language

It is a case insensitive language.

Every command must starts with verb.

ex: select, update, delete, drop and etc.

Every command must ends with semicolon.

It is developed by Mr.Codd in 1972 (By IBM).

Sub-Languages of SQL

We have five sub-language of SQL.

- 1) DDL (Data Definition Language)
- 2) DML (Data Manipulation Language)
- 3) DRL/DQL (Data Retrieve/Query Language)
- 4) TCL (Transaction Control Language)
- 5) DCL (Data Control Language)

1) DDL(Data Definition Language)

This language is used to maintain the objects in database.

It is a collection of five commands.

ex: create,alter,drop,truncate and rename.

2) DML(Data Manipulation Language)

This language is used to manipulate the data present in database.

It is a collection of four commands.

ex: insert,update,delete and merge.

3) DRL/DQL(Data Retrieve/Query Language)

This language is used to retrieve the data from database.

It is a collection of one command.

ex: select

4) TCL(Transaction Control Language)

This language is used to maintain the transaction of database.

It is a collection of three commands.

ex: commit,rollback and savepoint.

5) DCL(Data Control Language)

This language is used to control the access of data to the user.

It is a collection of two commands.

ex: grant and revoke.

Table

Table is an object which is used to represent the data.

Table is a collection of rows and columns.

Oracle is a case insensitive but data present in a table is a case sensitive.

ex: SNO | SNAME| SADD

SNO	SNAME	SADD
101	raja	hyd

102		ravi		delhi
103		ramana		vizag

Here we have 3 rows and 3 columns.

Oracle

Version	:	10g or 11g
Creator	:	Mr. Codd
Vendor	:	Oracle Corporation
Software	:	Expression Edition
website	:	www.oracle.com/in/database
Username	:	system (default)
Password	:	admin
Port No	:	1521

Download link:

https://drive.google.com/file/d/0B9rC21sL6v0td1NDZXpkUy1oMm8/view?usp=drive_link&resourcekey=0-aKooR3NmAh_eLo_qGw_inA

Establish the Connection with Database

To perform any operation on database we need to connect with database.

Once the work with database is completed then we need to disconnect with database.

There are following ways to connect and disconnect the database.

1)

SQL > connect

```
username: system
password: admin
```

SQL > disconnect

2)

SQL > conn

```
username: system
password: admin
```

SQL > disc

3)

SQL > conn system/admin

SQL > disc

create command

It is used to create a table(object) in a database.

syntax:

```
create table <table_name>(col1 datatype(size),col2 datatype(size),...,colN datatype(size));
```

ex: create table student(sno number(3),sname varchar2(10),sadd varchar2(12));
 create table dept(deptno number(3),dname varchar2(10),dloc varchar2(12));
 create table emp(eid number(3),ename varchar2(10), esal number(10,2),
 deptno number(3),job varchar2(10), comm number(6));

Describe command

It is used to describe the structure of the table.

syntax: desc <table_name>;

ex: desc student;

desc dept;

desc emp;

insert command

It is used to insert the data in a database table.

syntax: insert into <table_name> values(val1,val2,..,valN);

ex: insert into student values(101,'raja','hyd');
 insert into student values('raja',102,'delhi'); // invalid
 insert into student values(102,'ravi'); //invalid

A null is a keyword which represent undefined or unavailable.

insert into student values(102,'ravi',null);

Approach2

```
insert into student(sno,sname,sadd) values(103,'ramana','vizag');  
insert into student(sno,sname) values(105,'rakesh');
```

Approach3

Using '&' symbol we can read dynamic inputs.

ex: insert into student values(&sno,'&sname','&sadd');

commit command

It is used to make the changes permanent to database.

syntax: commit;

dept table

```
create table dept(deptno number(3),dname varchar2(10),dloc varchar2(12));
```

```
insert into dept values(10,'ECE','HYD');
insert into dept values(20,'CSE','PUNE');
insert into dept values(30,'EEE','VIZAG');
insert into dept values(40,'MEC','DELHI');
commit;
```

emp table

```
create table emp(eid number(3),ename varchar2(10),esal number(10,2),deptno number(3),
                job varchar2(10),comm number(6));
```

```
insert into emp values(201,'Alan',9000,10,'Clerk',null);
insert into emp values(202,'Kelvin',19000,10,'Clerk',500);
insert into emp values(203,'Nancy',29000,20,'HR',300);
insert into emp values(204,'Branda',18000,20,'HR',700);
insert into emp values(205,'Nelson',19000,30,'Manager',500);
insert into emp values(206,'Jose',49000,30,'Manager',200);
commit;
```

select command

It is used to select the records from database table.

syntax: select * from <table_name>; (Here '*' means all rows and columns.)

ex: select * from emp;
 select * from dept;
 select * from student;

Projection:

Selecting specific columns from the database table is called projection.

ex: select sno,sname,sadd from student;
 select sno,sname from student;
 select sno from student;

In select command we can perform arithmetic operations also.

ex: select sno+100 from student;
 select sno-100 from student;
 select sno*10 from student;

Column alias

A userdefined heading given to a column is called column alias.

Column alias can be applied to any column.

Column alias are temporary , once the query is executed we will loss the column alias.

ex: select sno+100 as SNO from student;

```
select sno as roll_no ,sname as Name,sadd as City from student;
```

Interview Queries

Q) Write a query to display all the tables present in database?

```
select * from tab;
```

Q) Write a query to display logical database name?

```
select * from global_name; // XE  
// ORCL
```

Q) Write a query to display employee information from emp table?

```
select * from emp;
```

Q) Write a query to display employee id, employee name and employee salary from emp table?

```
select eid,ename,esal from emp;
```

Q) Write a query to display employee id, employee name, employee salary and annual salary from emp table?

```
select eid,ename,esal,esal*12 from emp;
```

Q) Write a query to display employee id, employee name, employee salary and annual salary as ANNUAL_SAL from emp table?

```
select eid,ename,esal,esal*12 as ANNUAL_SAL from emp;
```

Where clause

Where clause is used to select specific records from database table.

syntax: select * from <table_name> where condition;

ex: select * from student where sno=101;

```
select * from student where sname='ravi';
```

```
select * from student where sadd='pune';
```

Interview Queries

Q) Write a query to display employees information those who are working in 10 department?

```
select * from emp where deptno=10;
```

Q) Write a query to display employees information whose who are working as a Manager?

```
select * from emp where job='Manager';
```

Q) Write a query to display employees information whose commission is null?

```
select * from emp where comm is null;
```

Update command

Update command is used to modify the data present in a database table.

syntax: update <table_name> set <column_name>=value where condition;

ex: update student set sname='rani' where sno=101;

update student set sname='alan',sadd='noida' where sno=102;

Note: If we won't use where clause then all rows will be updated.

ex: update student set sname='raja';
commit;

Delete command

Delete command is used to delete the records from database table.

syntax: delete from <table_name> where condition;

ex: delete from student where sno=101;

delete from student where sname='ravi';

delete from student where sadd is null;

Note: If we won't use where clause then all rows will be deleted.

ex: delete from student;
delete from dept;
delete from emp;
commit;

Note: ALL DML operations are temporary.

Interview Queries

Q) Write a query to increment salary of a employee by 1000 whose employee id is 201?

update emp set esal=esal+1000 where eid=201;

Q) Write a query to promote employee from clerk to salesman whose employee id is 202?

update emp set job='salesman' where eid=202;

Q) Write a query to terminate employees whose salary is greater than 30000?

delete from emp where esal>30000;

Q) Write a query to delete student record who is living in hyderabad?

delete from student where sadd='hyd';

Logical Operators

Logical operators are used to declare multiple conditions in where clause.

We have three logical operators.

1) AND

2) OR

3) NOT

1) AND

It will return the records if all the conditions are true.

In AND operator all conditions must be from same row.

ex: select * from emp where ename='Alan' AND esal=9000;
 select * from emp where ename='Alan' AND esal=19000; // no rows selected

2) OR

It will return the records if any condition is true.

In OR operator conditions can be from any row.

ex: select * from emp where ename='Alan' OR esal=9000;
 select * from emp where ename='Alan' OR esal=19000;
 select * from emp where ename='Alan' OR esal=79000;

3) NOT

It will return all the records except the condition.

A '<>' symbol denoted as NOT operator.

ex: select * from student where NOT sno=101;
 select * from student where sno<>101;
 select * from student where NOT sname='raja';
 select * from student where sname<>'raja';

Interview Queries

Q) Write a query to display employees information whose employee id is 203 and working as HR?

select * from emp where eid=203 AND job='HR';

Q) Write a query to display employee information whose employee id is 201,202 and 203?

select * from emp where eid=201 OR eid=202 OR eid=203;

Q) Write a query to display employees information whose salary is greater than 15000 and less than 30000?

select * from emp where esal>15000 and esal<30000;

Q) Write a query to display employees information those who are not working in 10 department?

select * from emp where NOT deptno=10;
select * from emp where deptno<>10;

Between operator

Between operator will return the records those who are in the range of values.

Between operator will take the support of AND operator.

In Between operator first we need to declare lower limit then higher limit.

ex: select * from emp where eid between 201 AND 206;
select * from emp where eid between 206 AND 201; // no rows selected
select * from emp where esal between 15000 AND 30000;
select eid,ename,esal from emp where deptno between 10 AND 30;

IN operator

IN operator is a replacement of OR operator.

It will return the values those who are matching in the list of values.

ex: select * from emp where eid IN(201,202,203);
select * from emp where deptno IN(10,20,30);
select * from emp where ename IN ('Alan','Jose','Nelson');

Pattern Matching operators

Pattern matching operators are used to select the letters from database table.

Pattern matching operators will take the support of like keyword.

We have two types of pattern matching operators.

- 1) Percentage(%)
- 2) Underscore(_)

1) Percentage(%)

Q) Write a query to display employee information whose employee name starts with 'A' letter?
select * from emp where ename like 'A%';

Q) Write a query to display employee information whose employee name ends with 'n' letter?

select * from emp where ename like '%n';

Q) Write a query to display employee information whose employee name having 'T' as middle letter?

select * from emp where ename like '%l%';

2) Underscore(_)

Q) Write a query to display employee information whose employee name having second letter as 'T' letter?

select * from emp where ename like '_l%';

Q) Write a query to display employee information whose employee name having second last letter as 'd'?

select * from emp where ename like '%d_';

Q) Write a query to display employee information whose employee name having third letter as 'a'?

```
select * from emp where ename like '__a%';
```

Duplicate table or Copy of a table

Creating a duplicate table or copy of a table is a save side for the programmer because if something goes wrong we can recover that situation using duplicate table or copy of a table.

Using create and select command we can create duplicate table.

ex: create table stud as select * from student;
 create table employee as select * from emp;
 create table employee as select eid,ename,esal from emp;
 create table employee as select * from emp where deptno=10;
 create table employee as select * from emp where deptno<>10;
 create table employee as select * from emp eid IN(201,202,203);
 create table employee as select * from emp esal between 15000 AND 30000;
 create table employee as select * from emp name like 'A%';

cl scr

It is used to clear the output screen of SQL command prompt.

syntax: cl scr

DDL commands

- create - (tables)
- alter - (columns)
- drop - (tables)
- truncate - (rows/records)
- rename - (tables)

alter command

Using alter command we can perform following activities very easily.

- i) Adding the new columns
- ii) Modifying the exiting columns
- iii) Renaming the columns
- iv) Droping the columns

i) Adding the new columns

Using alter command we can add new columns to a existing table.

syntax: alter table <table_name> ADD (col datatype(size));

ex: alter table student ADD (state varchar2(10));
 alter table student ADD (pincode number(8));

```
alter table student ADD (state varchar2(10), pincode number(8));
update student set state='Telangana' where sno=101;
```

ii) Modifying the exiting columns

Using alter command we can modify the existing columns.

We can change size of a column only when existing values are fit into new size.

```
ex:    desc student;
       alter table student MODIFY (state varchar2(12));
       desc student;
```

We can change the datatype of a column only if that column is empty.

```
ex:    desc student;
       alter table student MODIFY (pincode varchar2(8));
       desc student;
```

iii) Renaming a columns

Using alter command we can rename a column name.

syntax: alter table <table_name> RENAME column <old_name> to <new_name>

```
ex:    alter table student RENAME column pincode to country;
       alter table student RENAME column saddr to city;
       alter table emp RENAME column esal to dailywages;
       alter table emp RENAME column job to designation;
```

iv) dropping the columns

Using alter command we can drop the columns.

syntax: alter table <table_name> DROP (col1,col2,..,colN);

```
ex:    alter table student DROP (state,country);
```

drop command

It is used to drop the tables from database.

syntax: drop table <table_name>;

```
ex:    drop table student;
       drop table emp;
       drop table dept;
```

truncate command

A truncate command is used to delete the records permanently from database table.

syntax: truncate table <table_name>;

```
ex:    truncate table emp;
       truncate table student;
```

truncate table dept;

Q) What is the difference between delete and truncate command?

delete

It will delete the records temporary.

We can rollback the data.

Where clause can be used.

It is a DML command.

truncate

It will delete the records permanently.

We can't rollback the data.

Where clause can't be used.

It is a DDL command.

rename command

It is used to rename the table name.

syntax: rename <old_name> to <new_name>;

ex: rename emp to employee;

 rename student to students;

 rename dept to department;

Functions

Functions are used to manipulate the data items and give the result.

We have two types of functions.

1) Group Functions / Multiple Rows Functions

2) Scalar Functions / Single Rows Functions

1) Group Functions

Group functions are applicable for multiple rows.

We have following list of group functions.

ex: sum(), avg(), max(), min(), count(*) and count(express).

Q) Write a query to display sum of salary of each employee?

select sum(esal) from emp;

Q) Write a query to display average salary of each employee?

select avg(esal) from emp;

Q) Write a query to display highest salary from employee table?

select max(esal) from emp;

Q) Write a query to display least/lowest salary from employee table?

select min(esal) from emp;

Q) What is the difference between count(*) and count(exp)?

count(*)

It will return number of records present in database table.

It will include null records.

ex: select count(*) from emp;

count(exp)

It will return number of values present in a column.

It will not include null values.

ex: select count(eid) from emp;

select count(comm) from emp;

Userlist table

```
drop table userlist;
create table userlist(uname varchar2(10),pwd varchar2(10));
insert into userlist values('raja','rani');
insert into userlist values('king','kingdom');
commit;
```

Q) Write a query to check username and password is valid or not?

select count(*) from userlist where uname='raja' AND pwd='rani'; // 1

select count(*) from userlist where uname='raja' AND pwd='rani2'; // 0

Dual table

Dual table is a dummy table which is used to perform arithmetic operations and to see the current system date.

Dual table contains one row and one column.

ex: select 10+20 from dual;

select 10*20 from dual;

select sysdate from dual;

select current_date from dual;

2) Scalar Functions

Scalar Functions are applicable for single row.

Scalar functions are divided into four types.

i) Character functions

ii) Number functions

iii) Date functions

iv) Conversion functions

i) Character functions

upper()

It will convert lowercase to uppercase.

ex: select upper('oracle') from dual;
select upper('oracle') as UPPER from dual;

lower()

It will convert uppercase to lowercase.

ex: select lower('ORACLE') from dual;

initcap()

It will convert initial letter to capital.

ex: select initcap('this is oracle class') from dual;

lpad()

It will pad the characters at left side.

ex: select lpad('oracle',10,'z') from dual; //zzzzoracle

rpad()

It will pad the characters at right side.

ex: select rpad('oracle',10,'z') from dual; //oraclezzzz

ltrim()

It will trim the characters from left side.

ex: select ltrim('zzoraclezz','z') from dual; // oraclezz

rtrim()

It will trim the characters from right side.

ex: select rtrim('zzoraclezz','z') from dual; // zzoracle

trim()

It will trim the characters from both the sides.

ex: select trim('z' from 'zzoraclezz') from dual; // oracle

concat()

It will concatenate the strings.

ex: select concat('mega','star') from dual;
select concat(concat('mega','star'),'chiru') from dual;

replace()

It will replace the character in a string.

ex: select replace(esal,0,9) from emp;

ii) Number functions

abs()

It will return absolute value.

ex: select abs(-10) from dual; //10
select abs(10) from dual; //10

`power(A,B)`

It will return power value.

ex: `select power(5,3) from dual;`

`sqrt()`

It will return exact square root value.

ex: `select sqrt(25) from dual; //5`

`select sqrt(26) from dual; //5.09`

`ceil()`

It will return ceil value.

ex: `select ceil(10.2) from dual; // 11`

`select ceil(56.7) from dual; // 57`

`floor()`

It will return floor value.

ex: `select floor(10.2) from dual; //10`

`select floor(10.9) from dual; //10`

`round()`

It will return nearest value.

ex: `select round(10.4) from dual; // 10`

`select round(10.5) from dual; //11`

`trunc()`

It will return decimal values.

ex: `select trunc(10.56) from dual; //10`

`select trunc(56.78) from dual; // 56`

`greatest()`

It will return greatest value.

ex: `select greatest(101,102,103) from dual;`

`least()`

It will return least value.

ex: `select least(101,102,103) from dual;//101`

Working with Date values

All database softwares support date values.

It is not recommended to store date values in the form of varchar2.

Every database supports different date patterns.

ex: oracle -- dd-MMM-yy

MySQL -- yyyy-MM-dd

emp1 table

`drop table emp1;`

`create table emp1(eid number(3),ename varchar2(10),edoj date);`

```
insert into emp1 values(101,'Alan','01-JAN-24');  
insert into emp1 values(102,'Jose','15-NOV-23');  
insert into emp1 values(103,'Nelson',sysdate);  
commit;
```

iii) Date functions

We have following list of date functions.

ADD_MONTHS()

It will add the months in a given date.

ex: select ADD_MONTHS(sysdate,6) from dual;

MONTHS_BETWEEN()

It will return number of months between two given dates.

ex: select MONTHS_BETWEEN('01-JAN-23','11-JAN-24') from dual; // -12.3
select ABS(MONTHS_BETWEEN('01-JAN-23','11-JAN-24')) from dual; // 12.3

NEXT_DAY()

It will return next date of a given day in a week.

ex: select NEXT_DAY(sysdate,'SUNDAY') from dual;
select NEXT_DAY(sysdate,'THURSDAY') from dual;

LAST_DAY()

It will return last date of a month.

ex: select LAST_DAY(sysdate) from dual;
select LAST_DAY('15-FEB-24') from dual;

iv) Conversion functions

It is used to convert from one type to another type.

ex: TO_CHAR()

We have two pseudo for TO_CHAR().

number to_char()

It will accept '9' in digits , dollar and euro symbols.

ex: select eid,ename,TO_CHAR(esal,'9,999') from emp;
select eid,ename,TO_CHAR(esal,'99,999') from emp;
select eid,ename,TO_CHAR(esal,'\$99,999') from emp;
select eid,ename,TO_CHAR(esal,'\$99,999') as ESAL from emp;

date to_char()

```
select TO_CHAR(sysdate,'dd-MM-yyyy') from dual;  
select eid,ename,TO_CHAR(sysdate,'dd-MM-yyyy') as DOJ from emp1;  
select TO_CHAR(sysdate,'month') from dual;  
select TO_CHAR(sysdate,'mon') from dual;  
select TO_CHAR(sysdate,'year') from dual;  
select TO_CHAR(sysdate,'yyyy') from dual;
```

```
select TO_CHAR(sysdate,'mm') from dual;
select TO_CHAR(sysdate,'dd') from dual;
select TO_CHAR(sysdate,'day') from dual;
select TO_CHAR(sysdate,'dy') from dual;
select TO_CHAR(sysdate,'HH:MI:SS') from dual;
select TO_CHAR(sysdate,'yyyy-MM-dd HH:MI:SS') from dual;
```

Group by clause

Group by clause is used to divide the rows into groups so that we can apply group functions.
A column which we used in select clause then same column must use in group by clause.

Q) Write a query to display sum of salary of each department?

```
select sum(esal),deptno from emp group by deptno;
```

Q) Write a query to display average salary of each job?

```
select avg(esal),job from emp group by job;
```

Q) Write a query to display highest salary of each department?

```
select max(esal),deptno from emp group by deptno;
```

Q) Write a query to display lowest salary of each job?

```
select min(esal),job from emp group by job;
```

Having clause

Having clause is used to filter the rows from group by clause.

Having clause must used after group by clause.

Q) Write a query to display sum of salary of each department whose salary is greater than 30000?

```
select sum(esal),deptno from emp group by deptno having sum(esal)>30000;
```

Q) Write a query to display average salary of each job where average salary is less than 40000?

```
select avg(esal),job from emp group by job having avg(esal)<40000;
```

Order by clause

It is used to arrange the rows in a table.

By default it will arrange in ascending order.

ex: select * from emp order by eid;
 select * from emp order by eid desc;
 select * from emp order by ename;
 select * from emp order by esal;

Q)Write a query to display sum of salary of each department whose salary is greater then 30000?

```
select sum(esal),deptno from emp group by deptno  
having sum(esal)>30000  
order by deptno;
```

Q)Write a query to display sum of salary of each department except 20 department whose salary is greater then 30000?

```
select sum(esal),deptno from emp where deptno<>20 group by deptno  
having sum(esal)>30000  
order by deptno;
```

Integrity Constraints

Constraints are the rules which are applied on the tables.

We have following list of constraints.

- 1) NOT NULL
- 2) UNIQUE
- 3) PRIMARY KEY
- 4) FOREIGN KEY
- 5) CHECK

Every constraint can be created two levels.

- i) Column level
- ii) Table level

1) NOT NULL

NOT NULL constraint does not accept null values.

It will accept duplicate values.

NOT NULL constraint can be created at column level.

column level

```
drop table student;  
create table student(sno number(3) NOT NULL,sname varchar2(10),sadd varchar2(12));  
insert into student values(101,'raja','hyd');  
insert into student values(101,'ravi','delhi');  
insert into student values(null,'ramana','vizag'); //invalid  
commit;
```

Note:

NOT NULL constraint can be created for multiple columns.

ex:

```
drop table student;  
create table student(sno number(3) NOT NULL,
```

```
        sname varchar2(10) NOT NULL,  
        sadd varchar2(12) NOT NULL);  
insert into student values(101,'raja','hyd');  
insert into student values(null,'ravi','delhi'); //invalid  
insert into student values(102,null,'vizag'); //invalid  
insert into student values(102,'ramana',null); //invalid  
commit;
```

2) UNIQUE

UNIQUE constraint does not accept duplicate values.

UNIQUE constraint will accept null values.

UNIQUE constraint can be created at column level and table level.

column level

```
drop table student;  
create table student(sno number(3) UNIQUE,sname varchar2(10),sadd varchar2(12));  
insert into student values(101,'raja','hyd'); // valid  
insert into student values(101,'ravi','delhi'); // invalid  
insert into student values(null,'ramana','vizag'); // valid  
commit;
```

Note: UNIQUE constraint can be created for multiple columns.

table level

```
drop table student;  
create table student(sno number(3),sname varchar2(10),sadd varchar2(12), UNIQUE(sno));  
insert into student values(101,'raja','hyd'); // valid  
insert into student values(101,'ravi','delhi'); // invalid  
insert into student values(null,'ramana','vizag'); // valid  
commit;
```

Note: UNIQUE constraint can be created for multiple columns.

ex:

```
drop table student;  
create table student(sno number(3), sname varchar2(10),  
                    sadd varchar2(12), UNIQUE(sno,sname,sadd));
```

3) Primary Key

Primary key is a combination of NOT NULL and UNIQUE constraint.

Primary key does not accept null values and duplicate values.

A table can have only one primary key.

Primary key can be created at column level and table level.

column level

```

drop table student;
create table student(sno number(3) primary key,sname varchar2(10),sadd varchar2(12));
insert into student values(101,'raja','hyd'); //valid
insert into student values(101,'ravi','delhi'); // invalid
insert into student values(null,'ramana','vizag'); //invalid
commit;

```

table level

```

drop table student;
create table student(sno number(3),sname varchar2(10),sadd varchar2(12),primary key(sno));
insert into student values(101,'raja','hyd'); //valid
insert into student values(101,'ravi','delhi'); // invalid
insert into student values(null,'ramana','vizag'); //invalid
commit;

```

4) Foreign Key

Foreign key is used to establish the relationship between two tables.

This relationship is called parent and child relationship or master and detailed relationship.

To establish the relationship between two tables.A parent table must have primary key or unique key and child table must have foreign key.

A primary key name may or may not match with foreign key but datatype must match.

A foreign key will accept only those values which are present in primary key.

A foreign key will accept duplicate and null values also.

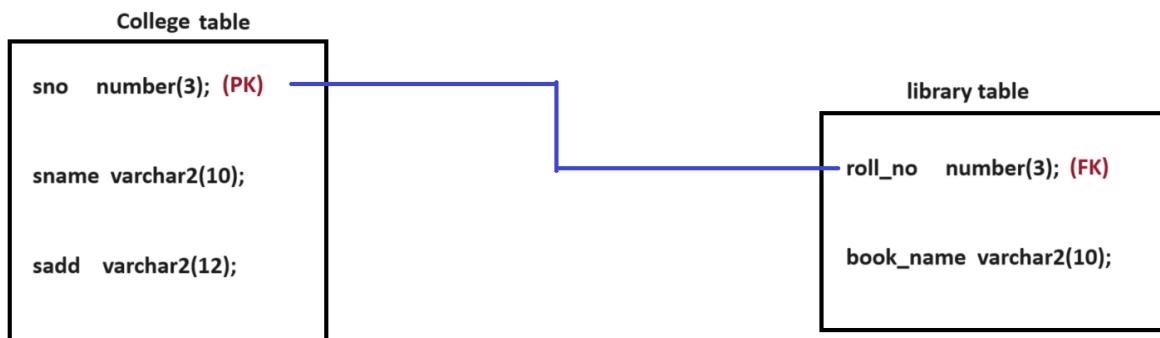


Diagram: oracle5.1

college table

```

drop table college;
create table college(sno number(3) primary key, sname varchar2(10), sadd varchar2(12));
insert into college values(101,'raja','hyd');
insert into college values(102,'ravi','delhi');
insert into college values(103,'ramana','vizag');
commit;

```

library table

```
drop table library;
create table library(roll_no number(3) REFERENCES college(sno), book_name varchar2(10));
insert into library values(101,'oracle');
insert into library values(102,'java');
insert into library values(103,'spring');
insert into library values(103,'reactjs');
insert into library values(null,'jdbc');
insert into library values(104,'servlets'); // invalid
commit;
```

5) CHECK

It is used to describe domain of a column.

Here domain means what type of values a column must accept.

Check constraint can be created at column level and table level.

column level

```
drop table student;
create table student(sno number(3),sname varchar2(10),smarks number(3)
CHECK(smarts<=100));
insert into student values(101,'raja',70);
insert into student values(102,'ravi',120); //invalid
insert into student values(103,'ramana',999); //invalid
commit;
```

ex:

```
drop table student;
create table student(sno number(3),sname varchar2(10), smarks number(3)
CHECK(smarks between 0 and 100));
insert into student values(101,'raja',70);
insert into student values(102,'ravi',120); //invalid
insert into student values(103,'ramana',999); //invalid
commit;
```

ex:

```
drop table student;
create table student(sno number(3), sname varchar2(10) CHECK(sname=lower(sname)),
smarks number(3));
insert into student values(101,'raja',70);
insert into student values(102,'RAVI',120); //invalid
```

```
insert into student values(103,'RaMaNa',999); //invalid  
commit;
```

ex:

```
drop table student;  
create table student(sno number(3), sname varchar2(10) CHECK(sname=upper(sname)),  
smarks number(3));
```

```
insert into student values(101,'RAJA',70);  
insert into student values(102,'ravi',120); //invalid  
insert into student values(103,'RaMaNa',999); //invalid  
commit;
```

table level

ex:

```
drop table student;  
create table student(sno number(3),sname varchar2(10), smarks number(3),  
CHECK(sname=upper(sname)));
```

```
insert into student values(101,'RAJA',70);  
insert into student values(102,'ravi',120); //invalid  
insert into student values(103,'RaMaNa',999); //invalid  
commit;
```

Note: Using constraints we can achieve accuracy and quality of data.

Q) How can we add constraint to a existing table?

```
alter table emp ADD primary key(eid);
```

Q) How can we remove constraint from existing table?

```
alter table emp DROP primary key;
```

TCL commands

- 1) commit
- 2) rollback
- 3) savepoint

1) commit

It is used to make the changes permanent to database.

ex: drop table student;

```
create table student(sno number(3),sname varchar2(10),sadd varchar2(12));  
insert into student values(101,'raja','hyd');  
insert into student values(102,'ravi','delhi');  
commit;
```

2) rollback

It is used to undo the changes which are not permanent.

```
syntax:      rollback;  
ex:   drop table student;  
create table student(sno number(3),sname varchar2(10),sadd varchar2(12));  
insert into student values(101,'raja','hyd');  
insert into student values(102,'ravi','delhi');  
commit;  
insert into student values(103,'ramana','vizag')  
insert into student values(104,'ramulu','pune');  
select * from student; //4 records  
rollback;  
select * from student; //2 records
```

3) savepoint

It is used to make logical transaction in a database table.

Instead of complete rollback we can rollback upto savepoint.

```
syntax:      savepoint <save_point_name>;  
ex:   drop table student;  
create table student(sno number(3),sname varchar2(10),sadd varchar2(12));  
insert into student values(101,'raja','hyd');  
insert into student values(102,'ravi','delhi');  
savepoint sp1;  
insert into student values(103,'ramana','vizag');  
insert into student values(104,'ramulu','pune');  
savepoint sp2;  
insert into student values(105,'Jack','USA');  
insert into student values(106,'James','UK');  
select * from student; // 6 records  
rollback to sp2;  
select * from student; // 4 records  
rollback to sp1;  
select * from student; // 2 records
```

DCL commands

- 1) grant
- 2) revoke

Schema: Schema is a memory location which is used to run SQL commands.

Privileges: Permissions given to a user is called privileges.

In general, rights given to a user is called privileges.

We have two types of privileges.

- 1) System privilege : Permissions given by DBA to user.
- 2) Object privilege : Permissions given by one user to another user.

1) grant

It is used to grant the permissions to the user.

syntax: grant <resource1>,<resource2> to <user_name>;

2) revoke

It is used to revoke the permissions from the user.

syntax: revoke <resource1>,<resource2> from <user_name>;

DBA> create user aravind identified by aravind;

DBA> create user manasa identified by manasa;

Aravind> conn aravind/aravind --logon denied

Manasa> conn manasa/manasa --logon denied

DBA> grant connect,resource to aravind,manasa;

Aravind> conn aravind/aravind

Manasa> conn manasa/manasa

Aravind>

create table employee(eid number(3),ename varchar2(10),esal number(10));

insert into employee values(201,'Alan',10000);

insert into employee values(202,'Jose',20000);

insert into employee values(203,'Mark',30000);

commit;

select * from employee;

Manasa> select * from employee; //table or view does not exist

Aravind> grant select on employee to manasa;

Manasa> select * from aravind.employee;

Manasa> delete from aravind.employee; --insufficient privileges

Aravind> grant delete,update on employee to manasa;

Manasa> delete from aravind.employee;

Manasa> commit;

```
Aravind> select * from employee; // no rows selected
```

```
Aravind> revoke select,update,delete on employee from manasa;
```

```
Manasa> disc
```

```
Aravind> disc
```

```
DBA> revoke connect,resource from aravind,manasa;
```

Pseudo Columns

A pseudo column means a column which is not real.

We have two pseudo columns.

- 1) ROWNUM
- 2) ROWID

1) ROWNUM

ROWNUM values always starts with 1 and increment by 1.

ROWNUM values are temporary. Once the query is executed we will loss the rownum values.

ex: select eid,ename,esal from emp;
 select rownum,eid,ename,esal from emp;
 select rownum,sno,sname,sadd from student;

2) ROWID

ROWID is a memory address where our records will store in a database table.

ROWID is permanent.

ex: select rowid,eid,ename,esal from emp;
 select rowid,rownum,eid,ename,esal from emp;

Interview Queries

Q) Write a query to see the list of users present in database?

```
select username from all_users;
```

Q) Write a query to drop the user?

```
drop user aravind cascade;  
drop user manasa cascade;
```

Q) Write a query to display first three records from employee table?

```
select * from emp where rownum<=3;
```

Q) Write a query to display exact fourth record from employee table?

```
select * from emp where rownum<=4
```

minus

```
select * from emp where rownum<=3;
```

Synonyms

Alternate name given to a table is called synonym.

We can use synonym name instead of table name for all the commands.

Using synonym, length of the table will reduce.

syntax: create synonym <synonym_name> for <table_name>;

ex: create synonym sy1 for student;

```
select * from sy1;
```

```
delete from sy1;
```

Q) Write a query to the list of synonyms present in database?

```
select synonym_name from user_synonyms;
```

Q) Write a query to drop the synonyms?

```
drop synonym sy1;
```

Sequence

A sequence is an object which is used to generate the numbers.

syntax: create sequence <sequence_name> start with value increment by value;

ex: create sequence sq1 start with 101 increment by 1;

```
create sequence sq2 start with 10 increment by 10;
```

We have two pseudo in a sequence.

1) NEXTVAL

It is used to generate next number in a sequence.

ex: create sequence sq1 start with 101 increment by 1;

```
drop table student;
```

```
create table student(sno number(3),sname varchar2(10),sadd varchar2(12));
```

```
insert into student values(sq1.NEXTVAL,'raja','hyd');
```

```
insert into student values(sq1.NEXTVAL,'ravi','delhi');
```

```
insert into student values(sq1.NEXTVAL,'ramana','vizag');
```

```
commit;
```

2) CURRVAL

It will return the last number which is generated by sequenced.

ex: select sq1.CURRVAL from dual;

Q) Write a query to display list of sequences present in database?

```
select sequence_name from user_sequences;
```

Q) Write a query to drop the sequence from database?

```
drop sequence sq1;
```

Indexes

Index is used to improve the performance of select command

Index in a database is same as index in a book.

Index can be created only to those columns which are widely used in where clause.

Whenever we create index, two columns will be generated. one is ROWID and another is indexed column. All the records will store in ascending order in a indexed column.

ex:

Indexed table

ROWID	INDEXED_COLUMN
	9000
	13000
	23000
	28000
	49000

We have two types of indexes.

1) Simple Index

If index is created for one column is called simple index.

ex: create index idx1 ON emp(esal);

Here index is used when we use esal in where clause.

ex: select * from emp where esal=49000;

2) Complex Index

If index is created for multiple columns is called complex index.

ex: create index idx2 ON emp(eid,deptno);

Here index is used when we use eid and deptno in where clause.

ex: select * from emp where eid=201 AND deptno=10;

Q) Write a query to see list of indexes present in database?

```
select index_name from user_indexes;
```

Q) Write a query to drop the indexes from database?

```
drop index idx1;
```

```
drop index idx2;
```

Joins

Joins are used to retrieve the data from one or more than one table.

ex: select * from emp,dept; //6*4 =24 records
 select eid,ename,esal,dname,dloc from emp,dept; //6*4 = 24 records
 select eid,ename,esal,deptno,dname,dloc from emp,dept;//column ambiguously defined

To overcome this limitation we need to use table_name.column_name.

Select emp.eid,emp.ename,emp.esal,dept.deptno,dept.dname,dept.dloc from emp,dept;//6*4=24

Table alias

A userdefined name given to a table is called table alias.

Table alias is temporary.

Once the query is executed we will loss the table alias.

Using table alias length of the query will reduce meanwhile performance will be maintained.

ex: select e.eid,e.ename,e.esal,d.deptno,d.dname,d.dloc from emp e,dept d; // 6*4=24 records

We have following types of joins.

- 1) Equi join
- 2) Non-Equi join
- 3) Self join
- 4) Cartisian Product
- 5) Inner join
- 6) Outer join

1) Equi join

When two tables are joined based on common column is called equi join.

ex: select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e,dept d
 where(e.deptno=d.deptno); // 6 records

2) Non-Equi join

When two tables are joined without equi join condition is called non-equi join.

ex: select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e,dept d
 where esal>25000; // 2 * 4 = 8 records

3) Self join

When table is joined to itself is called self join.

In self join we will create two table alias for the same table.

ex: select e1.eid,e1.ename,e1.esal,e2.job,e2.comm from emp e1,emp e2
 where(e1.deptno=e2.deptno); // 6 + 6 = 12 records

4) Cartisian product

When tables are joined without using any condition is called cartisian product.

It will return all the possible combinations.

ex: select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e,dept d; // $6 \times 4 = 24$ records

5) Inner join

Inner join is similar to equi join.

Inner join given by ANSI people

ANSI stands for American National standards Institute.

ex: select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e INNER JOIN dept d
ON(e.deptno=d.deptno); // 6 records

ex: select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e JOIN dept d
ON(e.deptno=d.deptno);

6) Outer join

It is a extension of equi join.

It will return matching as well as not matching records.

A '+' symbol denoted as outer join operator.

We have three types of outer joins.

i) Left outer join

SQL

```
select e.eid,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.dloc from emp e,dept d  
where(e.deptno=d.deptno(+));
```

ANSI

```
select e.eid,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.dloc  
from emp e LEFT OUTER JOIN dept d  
ON(e.deptno=d.deptno);
```

ii) right outer join

SQL

```
select e.eid,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.dloc from emp e,dept d  
where(e.deptno(+)=d.deptno);
```

ANSI

```
select e.eid,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.dloc  
from emp e RIGHT OUTER JOIN dept d  
ON(e.deptno=d.deptno);
```

iii) full outer join

ANSI

```
select e.eid,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.dloc
```

```
from emp e FULL OUTER JOIN dept d  
ON(e.deptno=d.deptno);
```

Merge command

Merge command is a combination of insert and update command.

student10 table

```
drop table student10;  
create table student10(sno number(3),sname varchar2(10),sadd varchar2(12));  
insert into student10 values(101,'raja','hyd');  
insert into student10 values(102,'ravi','delhi');  
insert into student10 values(103,'ramana','vizag');  
commit;
```

student20 table

```
drop table student20;  
create table student20(sno number(3),sname varchar2(10),sadd varchar2(12));  
insert into student20 values(103,'Alan','Texas');  
insert into student20 values(104,'James','Florida');  
commit;
```

ex:

```
merge into student10 s1  
using student20 s2  
ON(s1.sno=s2.sno)  
when matched  
then update set sname=s2.sname,sadd=s2.sadd  
when not matched  
then insert(sno,sname,sadd) values(s2.sno,s2.sname,s2.sadd);
```

View

View is a logical or virtual representation of a data from one or more than one table.

A table which is used to create a view is called base table or above table.

A view does not consumes the memory.

A view does not have any data.

A view will get the data when we execute select command.

syntax: create view <view_name> as select stmt;

ex: create view v1 as select * from emp;

 create view v1 as select eid,ename,esal from emp;

 create view v1 as select * from emp where eid IN(201,202,203);

```
create view v1 as select * from emp where ename like 'A%';
create view v1 as select * from emp where deptno<>10;
create view v1 as select * from emp where eid between 201 and 206;
```

We have following types of views.

- 1) Simple view
- 2) Complex view
- 3) With read only view
- 4) With check option view
- 5) Materialized view

1) Simple view

If a view is created by using one base table is called simple view.

In simple view DML operations are allowed.

```
ex:    create view v1 as select * from emp;
          select * from v1;
          select * from emp;
          delete from v1 where eid=207;
          select * from v1;
          select * from emp;
```

2) Complex view

If a view is created by using more then one base table is called complex view.

In complex view DML operations are not allowed.

```
ex:    create view v2 as select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e,dept d
          where(e.deptno=d.deptno);
          select * from v2;
          delete from v2;
```

3) With read only view

If a view is created by using one base table and DML operations are not required then we need to use with read only view.

```
ex:    create view v3 as select * from emp with read only;
          select * from v3;
          delete from v3;
```

4) With check option view

If a view is created by using one base table and DML operations are required only if our condition is true then we need to use with check option view.

```
ex:    create view v4 as select * from emp where deptno=30 with check option;
          insert into v4 values(207,'Lara',60000,50,'Salesman',500)//invalid
          insert into v4 values(207,'Lara',60000,30,'Salesman',500);
          select * from v4;
          select * from emp;
```

5) Materialized view

Materialized view is also known as snapshot.

To create materialized view a table must have primary key.

ex: alter table emp ADD primary key(eid);
create materialized view v5 as select * from emp;
select * from v5;
delete from v5 where eid=207; // data manipulation not possible
delete from emp where eid=207; //record deleted
select * from v5; // record available here
exec DBMS_SNAPSHOT.REFRESH('v5');
Here DBMS_SNAPSHOT is a package name.
Here REFRESH is a procedure name.

Q) Write a query to see the list of views present in database?

```
select view_name from user_views;
```

Q) Write a query to drop the view?

```
drop view v1;  
drop view v2;  
drop view v3;  
drop view v4;  
drop materialized view v5;
```

Sub Queries

If we declare a query inside another query such concept is called sub query.

Sub queries are used to select the records based on unknown values.

We have following list of sub queries.

- 1) Single row sub query
- 2) Multiple row sub query
- 3) Multiple column sub query

1) Single row sub query

When subquery returns only one row is called single row sub query.

Subquery can be nested upto 32 levels.

ex: SQL

```
select * from emp where eid=201;  
Subquery  
select * from emp where eid=(select eid from emp where ename='Alan');  
Here sub query will execute first then parent query.
```

ex:

SQL

```
select * from emp where eid=201 and ename='Alan';  
subquery  
    select * from emp where  
        eid=(select eid from emp where esal=9000)  
        and  
        ename=(select ename from emp where eid=201);
```

Q) Write a query to display second highest salary from emp table?

```
select max(esal) from emp where esal<(select max(esal) from emp);
```

Q) Write a query to display employees information whose salary is greater than Branda salary?

```
select * from emp where esal>(select esal from emp where ename='Branda');
```

Q) Write a query to delete duplicate records from a database table ?

```
delete from student where rowid NOT IN(select max(rowid) from student group by sno);
```

2) Multiple row sub query

If a sub query returns more than one row is called multiple row sub query.

To perform multiple row sub query we need to use multiple row operators.

We have three types of multiple row operators.

- 1) ANY
- 2) ALL
- 3) IN

1) **ANY** select * from emp where esal > ANY(select esal from emp where deptno=10);
 select * from emp where esal < ANY(select esal from emp where deptno=10);

2) **ALL** select * from emp where esal > ALL(select esal from emp where deptno=10);

3) **IN** select * from emp where esal IN (select esal from emp where deptno=10);
 select * from emp where NOT esal IN (select esal from emp where deptno=10);

3) Multiple column subquery

If a sub query returns more than one column is called multiple column sub query.

In multiple column sub query we need to use IN operator.

ex: select * from emp where(eid,ename,esal) IN (select eid,ename,esal from emp where eid=201);

```
select eid,ename,esal from emp where(eid,ename,esal)  
IN (select eid,ename,esal from emp where eid=201);  
select eid,ename,esal from emp where(eid,ename,esal)  
IN (select eid,ename,esal from emp);
```

PL/SQL

PL/SQL stands for Procedural and Structured Query Language.

It is a extension of SQL and having following features.

1. We can achieve programming features like control statements, LOOPS and etc.
2. It will reduce network traffic.
3. We can display our error messages by using the concept of exception handling.
4. We can perform related operations by using the concept of triggers.
5. It will compile and store PL/SQL program permanent to database for repeated execution.

PL/SQL block

A PL/SQL program is also known as PL/SQL block.

syntax:

```
DECLARE
-
-      -- Declaration section
-
BEGIN
-
-      -- Executable section
-
EXCEPTION
-
-      -- Exception section
-
END;
/
```

Declaration section

A declaration section is used to declare variables, exceptions , cursors and etc.

It is optional section.

Executable section

A executable section contains lines of code which are used to complete a table.

*It is a mandatory section.

Exception section

A exception section contains lines of code which are executed when exception is raised.

It is a optional section.

To see the output in PL/SQL we need to use below command.

ex: SQL> set serveroutput on

Q) Write a PL/SQL program to display Hello World?

```
BEGIN  
DBMS_OUTPUT.PUT_LINE('Hello World');  
END;  
/
```

*Note: Here DBMS_OUTPUT is a package name.

Here PUT_LINE is a procedure name.

Here '/' is used to submit the PL/SQL block into database.

Q) Write a PL/SQL program to perform sum of two numbers?

```
DECLARE  
A number;  
B number;  
C number;  
BEGIN  
A := 10;  
B := 20;  
C := A+B;  
DBMS_OUTPUT.PUT_LINE(C);  
END;  
/
```

Declaration and Initialization using single line.

```
DECLARE  
A number:=10;  
B number:=20;  
C number:=A+B;  
BEGIN  
DBMS_OUTPUT.PUT_LINE(C);  
END;  
/
```

Using '&' symbol we can provide dynamic values.

```
DECLARE
```

```
A number;  
B number;  
C number;  
BEGIN  
A := &a;  
B := &b;
```

```
C := A+B;  
DBMS_OUTPUT.PUT_LINE('sum of two numbers is ='||C);  
END;  
/
```

In PL/SQL , DML operations are allowed.

```
DECLARE  
L_Sno number(3);  
L_Sname varchar2(10);  
L_Sadd varchar2(12);  
BEGIN  
L_Sno := &sno;  
L_Sname := '&sname';  
L_Sadd := '&sadd';  
insert into student values(L_Sno,L_Sname,L_Sadd);  
DBMS_OUTPUT.PUT_LINE('Record Inserted');  
END;  
/
```

Q) Write a PL/SQL program to accept student number and update student name ?

```
DECLARE  
L_Sno number(3);  
BEGIN  
L_Sno := &sno;  
update student set sname='rani' where sno=L_Sno;  
DBMS_OUTPUT.PUT_LINE('Record Updated');  
END;  
/
```

Q) Write a PL/SQL program to accept student number and delete the record?

```
DECLARE  
L_Sno number(3);  
BEGIN  
L_Sno := &sno;  
delete from student where sno=L_Sno;  
DBMS_OUTPUT.PUT_LINE('Record Deleted');  
END;  
/
```

In PL/SQL , DML operations are allowed.

If we perform select command in PL/SQL then we need to use "into" clause.

Q) Write a PL/SQL program to display employee name whose employee id is 201?

```
DECLARE
L_Ename varchar2(10);
BEGIN
select ename into L_Ename from emp where eid=201;
DBMS_OUTPUT.PUT_LINE(L_Ename);
END;
/
```

Q) Write a PL/SQL program to display employee name, employee salary based on employee id?

```
DECLARE
L_Eid number(3);
L_Ename varchar2(10);
L_Esal number(10,2);
BEGIN
L_Eid := &eid;
select ename,esal into L_Ename,L_Esal from emp where eid=L_Eid;
DBMS_OUTPUT.PUT_LINE(L_Ename||' '||L_Esal);
END;
/
```

Percentage(%) TYPE Attribute

It is used to declare a local variable with respect to a column type.

syntax: variable_name table_name.column_name%TYPE;

Q) Write a PL/SQL program to display employee name, employee salary based on employee id?

```
DECLARE
L_Eid emp.eid%TYPE;
L_Ename emp.ename%TYPE;
L_Esal emp.esal%TYPE;
BEGIN
L_Eid := &eid;
select ename,esal into L_Ename,L_Esal from emp where eid=L_Eid;
DBMS_OUTPUT.PUT_LINE(L_Ename||' '||L_Esal);
END;
/
```

Percentage(&) ROWTYPE attribute

It is used to declare a local variable which holds complete row a table.

ROWTYPE variable can't display directly.

We can access ROWTYPE variables values by using variable_name.column_name.

syntax: variable_name table_name%ROWTYPE;

Q) Write a PL/SQL to display employee information whose employee id is 205?

```
DECLARE
A emp%ROWTYPE;
BEGIN
select * into A from emp where eid=205;
DBMS_OUTPUT.PUT_LINE(A.eid||' '||A.ename||' '||A.esal||' '||A.deptno||' '||A.comm||' '||A.job);
END;
/
```

Q)Write a PL/SQL to display employee information based on employee id?

```
DECLARE
L_Eid emp.eid%TYPE;
A emp%ROWTYPE;
BEGIN
L_Eid := &eid;
select * into A from emp where eid=L_Eid;
DBMS_OUTPUT.PUT_LINE(A.eid||' '||A.ename||' '||A.esal||' '||A.deptno||' '||A.comm||' '||A.job);
END;
/
```

*Note: To see the output in PL/SQL we need to use below command.

ex: SQL> set serveroutput on

Control Statements

We have three types of control statements in PL/SQL.

1. IF THEN
2. IF THEN ELSE
3. IF THEN ELSIF

1) IF THEN

It will evaluate the code only if our condition is true.

```
DECLARE
A number:=5;
BEGIN
DBMS_OUTPUT.PUT_LINE('welcome');
```

```
IF A>2 THEN
DBMS_OUTPUT.PUT_LINE('It is greatest');
END IF;
DBMS_OUTPUT.PUT_LINE('thankyou');
END;
/
o/p:
```

```
welcome
It is greatest
thankyou
```

ex: DECLARE
 A number:=5;
 BEGIN
 DBMS_OUTPUT.PUT_LINE('welcome');

 IF A>20 THEN
 DBMS_OUTPUT.PUT_LINE('It is greatest');
 END IF;

 DBMS_OUTPUT.PUT_LINE('thankyou');
 END;
 /
o/p:
 welcome
 thankyou

2) IF THEN ELSE

It will evaluate the code either our condition is true or false.

```
DECLARE
A number:=5;
BEGIN
DBMS_OUTPUT.PUT_LINE('welcome');

IF A>0 THEN
DBMS_OUTPUT.PUT_LINE('It is positive');
ELSE
DBMS_OUTPUT.PUT_LINE('It is negative');
```

```
END IF;
DBMS_OUTPUT.PUT_LINE('thankyou');
END;
/
o/p:
```

```
welcome
It is positive
thankyou
```

ex: DECLARE
 A number:=-5;
 BEGIN
 DBMS_OUTPUT.PUT_LINE('welcome');

 IF A>0 THEN
 DBMS_OUTPUT.PUT_LINE('It is positive');
 ELSE
 DBMS_OUTPUT.PUT_LINE('It is negative');
 END IF;

 DBMS_OUTPUT.PUT_LINE('thankyou');
 END;
 /
o/p:

```
welcome
IT is negative
thankyou
```

3) IF THEN ELSIF

It will evaluate the code based on multiple conditions.

```
DECLARE
opt number(3);
BEGIN
opt:=&opt;

IF opt=100 THEN
DBMS_OUTPUT.PUT_LINE('It is police number');
ELSIF opt=103 THEN
DBMS_OUTPUT.PUT_LINE('It is enquiry number');
ELSIF opt=108 THEN
```

```
DBMS_OUTPUT.PUT_LINE('It is emergency number');
ELSE
DBMS_OUTPUT.PUT_LINE('Invalid option');
END IF;
END;
/
```

LOOPS

We have three types of loops in PL/SQL.

1. Simple Loop
2. While Loop
3. For Loop

1) Simple Loop

It will evaluate the code until our condition is true.

```
DECLARE
A number:=1;
BEGIN
DBMS_OUTPUT.PUT_LINE('welcome');
```

```
LOOP
DBMS_OUTPUT.PUT_LINE('Hello');
EXIT WHEN A=4;
A:=A+1;
END LOOP;
```

```
DBMS_OUTPUT.PUT_LINE('thankyou');
END;
/
o/p:
```

welcome
Hello
Hello
Hello
Hello
thankyou

Q) Write a PL/SQL program to display 10 natural numbers?

```
DECLARE
A number:=1;
BEGIN
```

```
LOOP
DBMS_OUTPUT.PUT_LINE(A);
EXIT WHEN A=10;
A:=A+1;
END LOOP;

END;
/
```

2) While Loop

It will evaluate the code until our condition is true.

```
DECLARE
A number:=1;
BEGIN
DBMS_OUTPUT.PUT_LINE('welcome');

while A<=4 LOOP
DBMS_OUTPUT.PUT_LINE('Hello');
A:=A+1;
END LOOP;

DBMS_OUTPUT.PUT_LINE('thankyou');
END;
/
```

Q) Write a PL/SQL program to display 10 naturals in descending order?

```
DECLARE
A number:=10;
BEGIN

WHILE A>=1 LOOP
DBMS_OUTPUT.PUT_LINE(A);
A:=A-1;
END LOOP;

END;
/
```

3) For Loop

It will evaluate the code until our condition is true.

```
DECLARE
  A number;
BEGIN
  DBMS_OUTPUT.PUT_LINE('welcome');

  FOR A IN 1 .. 4 LOOP
    DBMS_OUTPUT.PUT_LINE('Hello');
  END LOOP;

  DBMS_OUTPUT.PUT_LINE('thankyou');
END;
```

/

o/p:

```
welcome
Hello
Hello
Hello
Hello
Thankyou
```

Q) Write a PL/SQL program to display multiplicate table of a given number?

```
DECLARE
  I number:=1;
  N number;
BEGIN
  N:=&n;
  FOR I IN 1 .. 10 LOOP
    DBMS_OUTPUT.PUT_LINE(N||' * '|I|| = '|N*I);
  END LOOP;
END;
/
```

Exceptions

Runtime errors are called exceptions.

We have two types of exceptions in PL/SQL.

- 1) Predefined Exceptions
- 2) Userdefined Exceptions

1) Predefined Exceptions

Built-In exceptions are called predefined exceptions.

We have following list of predefined exceptions.

- i) NO_DATA_FOUND Exception
- ii) TOO_MANY_ROWS Exception
- iii) VALUE_ERROR Exception
- iv) ZERO_DIVIDE Exception
- v) DUP_VAL_ON_INDEX Exception
- vi) OTHERS

i) NO DATA FOUND

This exception will occur when select statement does not return any record.

```
DECLARE
L_Ename emp.ename%TYPE;
BEGIN
select ename into L_Ename from emp where eid=209;
DBMS_OUTPUT.PUT_LINE(L_Ename);
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Please check employee id');
END;
/
```

ii) TOO MANY ROWS

This exception will raise when select statement returns more then one row.

```
DECLARE
L_Ename emp.ename%TYPE;
BEGIN
select ename into L_Ename from emp where deptno=10;
DBMS_OUTPUT.PUT_LINE(L_Ename);
EXCEPTION
WHEN TOO_MANY_ROWS THEN
DBMS_OUTPUT.PUT_LINE('select stmt returns more then one row');
END;
/
```

iii) VALUE_ERROR

This exception will raise when there is a mismatch with datatype or size.

```
DECLARE
A number(3);
BEGIN
```

```

A:=123456;
DBMS_OUTPUT.PUT_LINE(A);
EXCEPTION
WHEN VALUE_ERROR THEN
DBMS_OUTPUT.PUT_LINE('Please check the size');
END;
/

```

ex:2

```

DECLARE
L_Esal emp.esal%TYPE;
BEGIN
select ename into L_Esal from emp where eid=201;
DBMS_OUTPUT.PUT_LINE(L_Esal);
EXCEPTION
WHEN VALUE_ERROR THEN
DBMS_OUTPUT.PUT_LINE('Please check the datatype');
END;
/

```

iv) ZERO_DIVIDE

This exception will raise when we are trying to divide a number with zero.

```

DECLARE
A number;
BEGIN
A:=10/0;
DBMS_OUTPUT.PUT_LINE(A);
EXCEPTION
WHEN ZERO_DIVIDE THEN
DBMS_OUTPUT.PUT_LINE('Dont divide by zero ');
END;
/

```

v) DUP_VAL_ON_INDEX

This exception will raise when are trying to insert duplicate value in a primary key.

```

alter table student add primary key (sno);
BEGIN
insert into student values(101,'jose','florida');
DBMS_OUTPUT.PUT_LINE('Record inserted');
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN

```

```
DBMS_OUTPUT.PUT_LINE('Duplicate records not allowed');
END;
/
```

vi) OTHERS

It is a universal angular exception which handles all types of exceptions.

```
DECLARE
L_Ename emp.ename%TYPE;
BEGIN
select ename into L_Ename from emp where eid=209;
DBMS_OUTPUT.PUT_LINE(L_Ename);
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Please check employee id');
END;
/
```

2) Userdefined Exceptions

Exceptions are created by the user are called userdefined exceptions.

steps to develop user defined exceptions

- step1: Declare the exception
- step2: Raise the exception
- step3: Handle the exception

```
DECLARE
A number:=5000;
MY_EX Exception;
BEGIN

IF A>2000 THEN
RAISE MY_EX;
END IF;
DBMS_OUTPUT.PUT_LINE(A);

EXCEPTION
WHEN MY_EX THEN
DBMS_OUTPUT.PUT_LINE('Number is too large');
END;
/
```

Cursors

Cursor is a memory location which is used to run SQL commands.

We have two types of cursors.

- 1) Implicit cursor
- 2) Explicit cursor

1) Implicit cursor

All the activities related to cursor like opening the cursor, processing the cursor and closing the cursor which is done automatically is called implicit cursor.

We have four types of implicit cursor attributes.

i) SQL%ISOPEN

It is a boolean attribute value which always returns false.

ii) SQL%FOUND

It is a boolean attribute which returns true if SQL command is success and returns false if SQL command is failed.

iii) SQL%NOTFOUND

It is completely inverse of SQL%FOUND.

It is a boolean attribute which returns false if SQL command is success and returns true if SQL command is failed.

iv) SQL%ROWCOUNT

It will return number of records effected in a database table.

SQL%ISOPEN

```
BEGIN
  IF SQL%ISOPEN THEN
    DBMS_OUTPUT.PUT_LINE('Cursor is open');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Cursor is closed');
  END IF;
END;
/
```

SQL%FOUND

```
BEGIN
  update student set sname='rani' where sno=101;
  IF SQL%FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Record Updated');
  ELSE
```

```
DBMS_OUTPUT.PUT_LINE('Record Not Updated');
END IF;
END;
/
```

SQL%NOTFOUND

```
BEGIN
update student set sname='rani' where sno=105;
IF SQL%NOTFOUND THEN
DBMS_OUTPUT.PUT_LINE('Record Updated');
ELSE
DBMS_OUTPUT.PUT_LINE('Record Not Updated');
END IF;
END;
/
```

SQL%ROWCOUNT

```
BEGIN
update student set sname='raja';
DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT||' Records updated');
END;
/
```

2) Explicit cursor

All the activities related to cursor like open the cursor, processing the cursor and closing the cursor which is done by a programmer is called explicit cursor.

We will use explicit cursor when select statement returns more than one row.

We have four types of explicit cursor attributes.

i) %ISOPEN

It is a boolean attribute which returns true if cursor is opened and returns false if cursor is closed.

ii) %FOUND

It is a boolean attribute which returns true if SQL command is success and returns false if SQL command is failed.

iii) %NOTFOUND

It is a boolean attribute which returns false if SQL command is success and returns true if SQL command is failed.

iv) %ROWCOUNT

It will return number of records effected in a database table.

Steps to work with explicit cursor

step1: Declare the cursor

ex: CURSOR <cursor_name> is select stmt;

step2: Open the cursor

ex: OPEN <cursor_name>;

step3: Fetch the data from cursor to local variables

ex: FETCH <cursor_name> into <variables>;

step4: Close the cursor

ex: CLOSE <cursor_name>;

Q) Write a PL/SQL program to display employee names from emp table?

```
DECLARE
CURSOR C1 is select ename from emp;
L_ename emp.ename%TYPE;
BEGIN
OPEN C1;
LOOP
FETCH C1 into L_ename;
DBMS_OUTPUT.PUT_LINE(L_ename);
EXIT WHEN C1%NOTFOUND;
END LOOP;
CLOSE C1;
END;
/
```

Q)Write a PL/SQL program to display employee id , employee name and employee salary from emp table?

```
DECLARE
CURSOR C2 is select eid,ename,esal from emp;
L_eid emp.eid%TYPE;
L_ename emp.ename%TYPE;
L_esal emp.esal%TYPE;
BEGIN
OPEN C2;
LOOP
FETCH C2 into L_eid,L_ename,L_esal;
DBMS_OUTPUT.PUT_LINE(L_eid||' '||L_ename||' '||L_esal);
```

```
EXIT WHEN C2%NOTFOUND;
END LOOP;
CLOSE C2;
END;
/
```

Q) Write a PL/SQL program to display employees information emp table?

```
DECLARE
CURSOR C3 is select * from emp;
A emp%ROWTYPE;
BEGIN
OPEN C3;
LOOP
FETCH C3 into A;
DBMS_OUTPUT.PUT_LINE(A.eid||' '||A.ename||' '||A.esal||' '||A.deptno||' '||A.job||' '||A.comm);
EXIT WHEN C3%NOTFOUND;
END LOOP;
CLOSE C3;
END;
/
```

ex:

```
DECLARE
CURSOR C4 is select e.eid,e.ename,d.dname,d.dloc from emp e , dept d where
(e.deptno=d.deptno);
L_eid emp.eid%TYPE;
L_ename emp.ename%TYPE;
L_dname dept.dname%TYPE;
L_dloc dept.dloc%TYPE;
BEGIN
OPEN C4;
LOOP
FETCH C4 into L_eid,L_ename,L_dname,L_dloc;
DBMS_OUTPUT.PUT_LINE(L_eid||' '||L_ename||' '||L_dname||' '||L_dloc);
EXIT WHEN C4%NOTFOUND;
END LOOP;
CLOSE C4;
END;
/
```

Procedures

A procedure is a named PL/SQL block which compiled and execute in database for repeated execution.

It is also known as stored PL/SQL procedures.

syntax:

```
create or replace procedure <procedure_name>
is
begin
-
-
-
END;
/
```

Q) Write a procedure to display Hello World?

```
create or replace procedure p1
is
begin
DBMS_OUTPUT.PUT_LINE('Hello World');
END;
/
```

*Note: To execute the procedure we need to use below command.

ex: SQL> exec p1;

Every procedure may contains three parameters.

- 1) IN parameter
- 2) OUT parameter
- 3) IN OUT parameter

1) IN parameter

It will accept the values from the user.

Q) Write a procedure to perform sum of two numbers?

```
create or replace procedure sum(A IN number,B IN number)
is
C number;
begin
C:=A+B;
DBMS_OUTPUT.PUT_LINE('sum of two numbers is ='||C);
END;
/
```

We can execute the procedure by using below command.

ex: SQL> exec sum(10,20);

Using procedures we can perform DML operations.

Q) Write a procedure to update student name based on student number?

```
create or replace procedure update_name(L_sno IN student.sno%TYPE)
is
begin
update student set sname='rani' where sno=L_sno;
DBMS_OUTPUT.PUT_LINE('Record updated');
END;
/
```

To execute the procedure we need to use below command.

ex: SQL> exec update_name(101);

Q) Write a procedure to delete student record based on student number?

```
create or replace procedure delete_record(L_sno IN student.sno%TYPE)
is
begin
delete from student where sno=L_sno;
DBMS_OUTPUT.PUT_LINE('Record deleted');
END;
/
```

To execute the procedure we need to use below command.

ex: exec delete_record(102);

To see the output in PL/SQL we need to use below command.

ex: SQL> set serveroutput on

2) OUT parameter

It will return the output to the user.

Q) Write a procedure to perform sum of two numbers and return sum?

```
create or replace procedure ret_sum(A IN number,B IN number,C OUT number)
is
begin
C:=A+B;
END;
/
```

Steps to call a procedure having OUT parameter

step1: Declare a bind variable

ex: variable N number;

step2: Execute the procedure

ex: exec ret_sum(10,20,:N);

step3: Print a bind variable

ex: print N;

3) IN OUT parameter

It is used to accept and return a value from/to the user.

Q) Write a procedure to return square of a given number?

```
create or replace procedure ret_square(A IN OUT number)
is
begin
A := A*A;
END;
/
```

Steps to call a procedure having IN OUT parameter

step1: Declare a bind variable

ex: variable N number;

step2: Initialize the bind variable

```
ex: BEGIN
:N := 5;
END;
/
```

step3: Execute the procedure

ex: exec ret_square(:N);

step4: Print a bind variable

ex: print N;

Q) Write a query to see the list of procedures present in database?

```
select object_name from user_objects where object_type='PROCEDURE';
```

Q) Write a query to see the source code of a procedure?

```
select text from user_source where name='P1';
```

Q) Write a query to drop the procedure?

```
drop procedure p1;
```

```
drop procedure ret_sum;  
drop procedure ret_square;
```

PL/SQL functions

It is a named PL/SQL block which must and should returns a value.

syntax:create or replace function <function_name>

```
    return datatype  
    is  
    begin  
        -  
        -  
        -  
    end;  
    /
```

Q) Write a PL/SQL function to return sum of two numbers?

```
create or replace function f1(A number,B number)  
return number  
is  
C number;  
begin  
C:=A+B;  
return C;  
END;  
/
```

To execute the function we need to use below command.

```
ex:    select f1(10,20) from dual;  
        select f1(10,20) as SUM from dual;
```

Q) Write a PL/SQL function to accept one salary then find out 10% of tax?

```
create or replace function f2(SAL number)  
return number  
is  
TAX number;  
begin  
TAX := SAL*10/100;  
return TAX;  
END;  
/
```

To execute the function we need to use below command.

ex: select f2(10000) from dual;
select eid,ename,esal,f2(esal) as TAX from emp;

Note: In Functions DML operations are not allowed.

Q) Write a query to see the list of functions present in database?

select object_name from user_objects where object_type='FUNCTION';

Q) Write a query to see the source code of a function?

select text from user_source where name='F1';

Q) Write a query to drop the function?

drop function f1;
drop function f2;

Q) What is the difference between procedures and functions?

<u>Procedures</u>	<u>Functions</u>
Procedure may or may not returns a value.	Function always returns a value.
DML operations are allowed.	DML operations are not allowed.
Can't be invoked by using select stmt.	Can be invoked by using select stmt.

Packages

A package is a collection of logical related sub programs.

PL/SQL procedures and functions are called logical related sub programs.

Package creation involved in two steps.

1) Package specification

It contains declaration of logical related sub programs.

2) Package body

It contains definition of logical related sub programs.

ex:

package specification
create or replace package pkg1
is
procedure sum(A IN number,B IN number);
end pkg1;
/

package body

create or replace package body pkg1
is

```
procedure sum(A IN number,B IN number)
is
C number;
begin
C:=A+B;
DBMS_OUTPUT.PUT_LINE(C);
END;
end pkg1;
/
```

To call the procedure we need to use below command.

ex: exec pkg1.sum(10,40);

ex:

package specification

```
create or replace package pkg2
is
function ret_fun(A number,B number)
return number;
end pkg2;
/
```

package body

```
create or replace package body pkg2
is
function ret_fun(A number,B number)
return number
is
C number;
BEGIN
C:=A+B;
return C;
END;
end pkg2;
/
```

To call the function we need to use below command.

ex: select pkg2.ret_fun(10,20) from dual;

Q) Write a query to see the list of packages present in database?

```
select object_name from user_objects where object_type='PACKAGE';
```

Q) Write a query to see the source code of a package?

```
select text from user_source where name='PKG1';
```

Q) Write a query to drop the package?

```
drop package body pkg1;  
drop package pkg1;  
drop package body pkg2;  
drop package pkg2;
```

Triggers

Trigger is a PL/SQL block which is executed based on event.

Insert, Update and Delete are trigger events.

Before, After and Insteadof are the timings of trigger.

syntax: create or replace trigger <trigger_name> <timing> <event> ON <object>

```
begin
```

```
-
```

```
-
```

```
-
```

```
end;
```

```
/
```

ex: create or replace trigger trg1 after insert ON student

```
begin  
DBMS_OUTPUT.PUT_LINE('Thank you for inserting !!!');  
END;  
/  
select * from student;  
insert into student values(103,'ramana','vizag'); // trigger will execute
```

Trigger can be created for multiple events also.

ex: create or replace trigger trg2 before insert OR update OR delete ON student

```
begin  
IF inserting THEN  
DBMS_OUTPUT.PUT_LINE('Thank you for inserting!!');  
ELSIF updating THEN  
DBMS_OUTPUT.PUT_LINE('Thank you for updating!!');  
ELSE  
DBMS_OUTPUT.PUT_LINE('Thank you for deleting!!');  
END IF;
```

```
END;
/
select * from student; // no trigger
insert into student values(104,'ramulu','pune');
update student set sname='rani' where sno=104;
delete from student where sno=104;
```

Triggers are categories into two types.

1) Statement level trigger

In statement level trigger, trigger will execute only for one time irrespective of number of records effected in a database table.

By default every trigger is a statement level trigger.

```
create or replace trigger trg3 after delete on student
begin
DBMS_OUTPUT.PUT_LINE('Deleted!!');
END;
/
delete from student; // trigger will execute only for one time.
```

2) Row level trigger

In row level trigger, trigger will execute irrespective of number of records effected in a database table.

To create row level trigger we need to use "FOR EACH ROW" clause.

```
create or replace trigger trg4 after delete on student FOR EACH ROW
begin
DBMS_OUTPUT.PUT_LINE('Deleted!!');
END;
/
delete from student; // trigger will execute here multiple times
```

Q) Write a query to see the list of triggers present in database?

```
select object_name from user_objects where object_type='TRIGGER';
```

Q) Write a query to see the source code of a trigger?

```
select text from user_source where name='TRG4';
```

Q) Write a query to drop the trigger from database?

```
drop trigger trg1;
drop trigger trg2;
drop trigger trg3;
drop trigger trg4;
```

JDBC

JDBC

As if know it is known as trademark.

But earlier it is called as Java Database Connectivity.

RAM is a temporary storage device or medium.

During the program execution our data will store in RAM.

Once the program execution is completed we will loss the data.

To overcome this limitation we are making our application writing the data into file or database software.

Files and Database software's act like a permanent storage device or medium.

Persistence

The process of storing and managing the data for a long period of time is called persistence.

Important Terminologies

1) Persistence Store

It is a place where we can store and manage the data for a long period of time.

ex:

Database software's

Files

2) Persistence Data

Data of a persistence store is called persistence data.

ex:

tables / collections

records

3) Persistence operation

Insert, update, delete, create and etc are called persistence operation.

In the real-time this operation is also known as CURD operation, CRUD operation, SCUD operation.

ex:

C - create

S - select

U - update

C - create

R - read

U - update

D - delete

D - delete

4) Persistence logic

A logic which is capable to perform persistence operations is called persistence logic.

ex:

JDBC code

Hibernate code

IOStream

5) Persistence technology

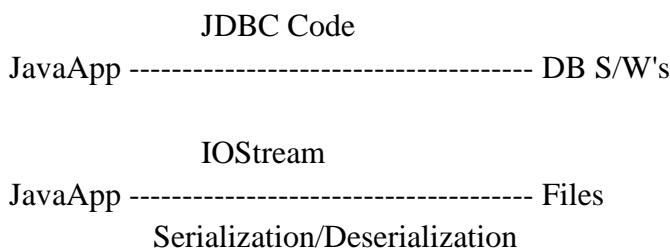
A technology which is used to develop persistence logic is called persistence technology.
ex:

JDBC
Hibernate
JPA
and etc.

Q) What is JDBC?

JDBC is a persistence technology which is used to develop persistence logic having the capability to perform persistence operations on persistence data of a persistence store.

Note:



Serialization

The process of storing object data into a file is called serialization.

In serialization, object will not store in a file instead object data will store in a file.

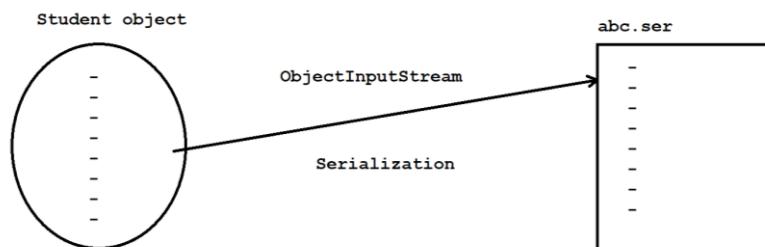


Diagram: jdbc1.1

In general, Serialization means converting object state to file state.

Deserialization

The process of taking the data from a file and represent an object is called deserialization.

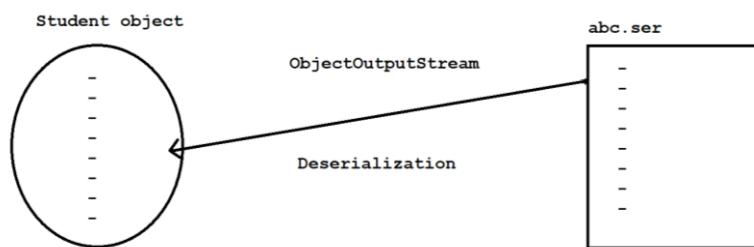


Diagram: jdbc1.2

Limitations with Files as persistence store

- > It will store limited amount of data.
- > There is no security.
- > Fetching the data with multiple conditions is not possible.
- > It does not show an application with relationship.
- > It does not allow us to apply constraints (primary key, unique key, not null).
- > Updating and Deletion of data can't be done directly.
- > Merging and comparison of data can't be done easily.

Advantages of Database software as persistence store

- > We can store unlimited amount of data.
- > There is a security.
- > It supports common query language.
- > Fetching the data with multiple conditions is possible.
- > It shows an application with relationships.
- > It allows us to apply constraints.
- > Deletion and Updating of data can be done directly.
- > Merging and comparison of data can be done easily.

Every JDBC application is a two-tier application where java with JDBC code acts like a frontend/tier1/layer1 and database software acts like a backend/tier2/layer2.

Enduser is a non-technical person. He can't prepare and execute SQL query in database software. So he depends upon frontend developers having the capability to do that work for him.

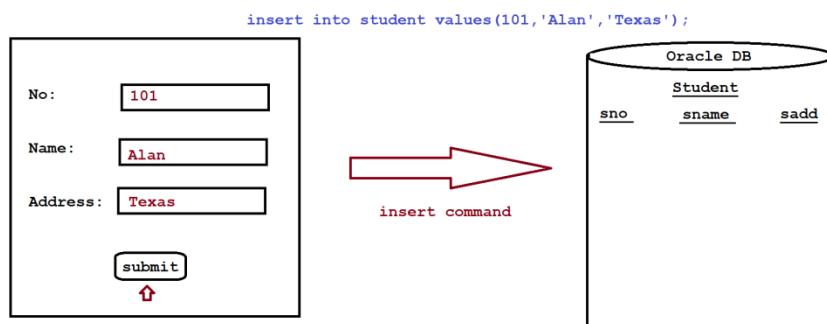


Diagram: jdbc1.3

JDBC Driver

It acts like a bridge between java application and database software.

It is used to convert Java calls to database calls and vice versa.

Here calls means instructions.

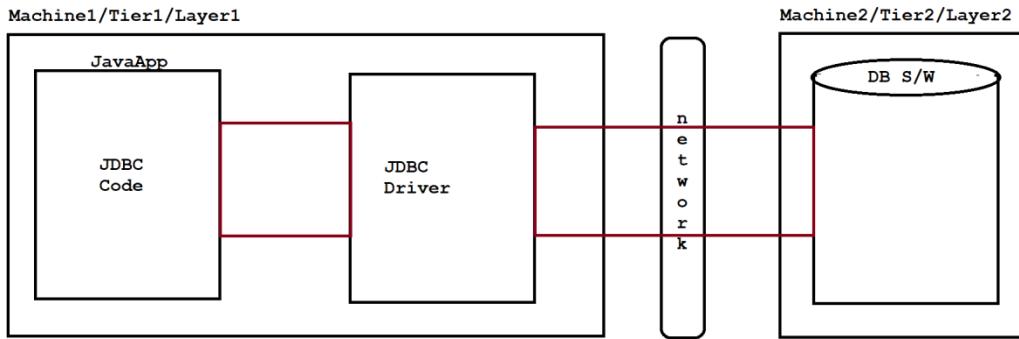


Diagram: jdbc1.4

ODBC Driver

VBScript, Perl, D2k and etc uses ODBC driver to interact with database software.

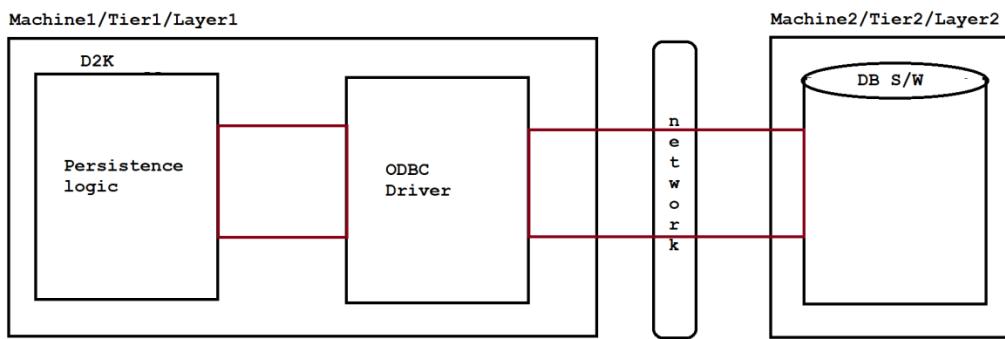


Diagram: jdbc1.5

ODBC driver is developed by using c language by taking the support of pointers. But Java does not support pointers. To overcome this limitation Sun Micro System introduced JDBC driver exclusively.

We will get JDBC software's from three parties.

- 1) Sun Micro System (creator of JDBC driver)
- 2) Database vendor
- 3) Third party vendor

We will get ODBC software's from three parties.

- 1) Xopen company (creator of ODBC driver)
- 2) Database vendor
- 3) Third party vendor

Q) What is JDBC?

It is a open technology given by Sun Micro System having set of rules and guidelines to develop JDBC drivers for multiple database software's.

Q) What is ODBC?

ODBC is a open technology given by Xopen company having set of rules and guidelines to develop ODBC drivers for multiple database software's.

Oracle

Version	:	10g or 11g
Vendor	:	Oracle corporation
Opensource	:	open source
Port No	:	1521
Username	:	system (default)
Password	:	admin
Website	:	https://www.oracle.com/in/database/

Download link :

https://drive.google.com/file/d/0B9rC21sL6v0td1NDZXpkUy1oMm8/view?usp=drive_link&resourcekey=0-aKooR3NmAh_eLo_qGw_inA

To use any JDBC driver we need to register with DriverManager service.

Every JDBC application contains one built-in service called DriverManager service.

Class.forName()

It always recommended to use Class.forName() method to register JDBC driver with DriverManager service.

It is used to load the driver class but it won't create an object.

ex:

```
Class.forName("driver-class-name");
```

Connection object

A Connection is an interface which is present in java.sql package.

It is an object of underlying supplied java class which implements java.sql.Connection interface.

If we want to interact with database we need to establish the connection with database software.

Once the work with database is completed, we need to close the Connection object.

ex:

```
Connection con;
```

DriverManager.getConnection()

A DriverManager is a class which is present in java.sql package.

A getConnection() static method is used to interact with database software and returns JDBC Connection object representing connectivity between java application and database software.

ex:

```
Connection con=DriverManager.getConnection("driver-url",username,pwd);
```

Statement object

A Statement is an interface which is present in java.sql package.

It acts like a vehicle between java application and database software.

It is used to sends and executes SQL query in database software.

We can create Statement object as follow.

ex:

```
Statement st=con.createStatement();
```

ResultSet object

A ResultSet is an interface which is present in java.sql package.

Every ResultSet contains two positions.

- 1) BFR (Before First Record/Row)
- 2) ALR (After Last Record/Row)

Bydefault record pointer points to BFR position.

Every record ResultSet having 1 as base index and every record ResultSet having 1 as column index.

rs.next()

It will move record pointer to next position from current position.

If next position is a record then it will return true.

If next position is ALR then it will return false.

To read the values from record ResultSet we need to use getXxx() method with index number or column name.

Here getXxx() method means getInt(),getFloat(),getDouble() and etc.

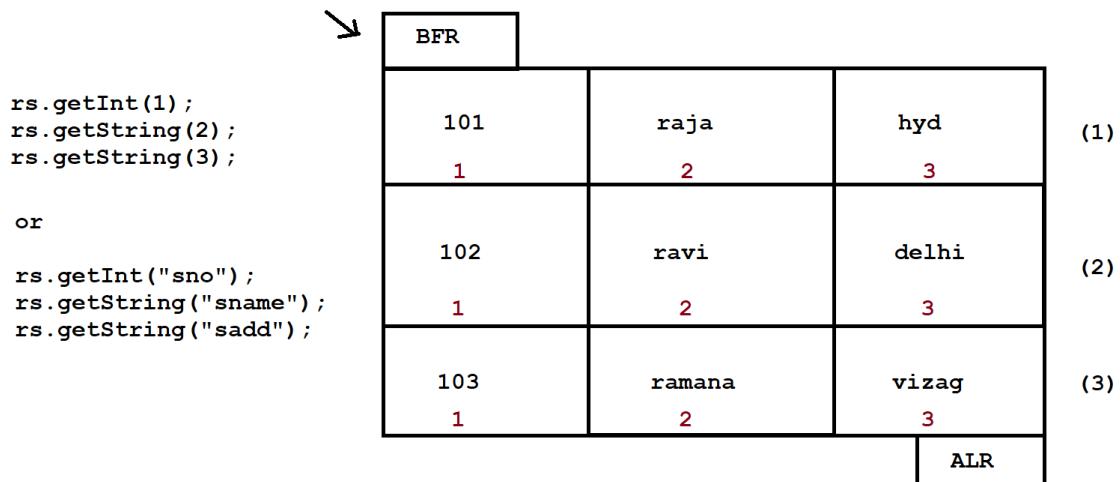


Diagram: jdbc2.1

student table

```
drop table student;
```

```
create table student(sno number(3),sname varchar2(10),sadd varchar2(12));
```

```
insert into student values(101,'raja','hyd');  
insert into student values(102,'ravi','delhi');  
insert into student values(103,'ramana','vizag');
```

```
commit;
```

```
select * from student;
```

Q)Difference between Oracle and MongoDB?

Oracle

It is a RDBMS database
table
row
column

MongoDB

MongoDB is a document base database.
Collection
document
field

Types of Queries in JDBC

According JDBC point of view we have two types of queries.

- 1)Select Query
- 2)Non-Select Query

1)Select Query

A select query will give bunch of records from database.

ex:

```
select * from student;
```

A JDBC Statement object gave executeQuery() method to execute select query.

ex:

```
ResultSet rs=st.executeQuery("select * from student");
```

2)Non-Select Query

A non-select query will return numeric value represent number of records effected in a database table.

ex:

```
delete from student;
```

A JDBC Statement object gave executeUpdate() method to execute non-select query.

ex:

```
int result=st.executeQuery("delete from student");
```

Steps to develop JDBC application

We have six steps to develop JDBC application.

- 1) Register JDBC driver with DriverManager service.
- 2) Establish the connection with database software.
- 3) Create Statement object.
- 4) Sends and Executes SQL query in database software.
- 5) Gather the result from database software to process the result.
- 6) Close all JDBC connection objects.

Q) How many drivers are there in JDBC?

We have four drivers in JDBC.

- Type1 JDBC driver / JDBC-ODBC bridge driver
- Type2 JDBC driver / Native API
- Type3 JDBC driver / Net Protocol
- Type4 JDBC driver / Native Protocol

Type4 JDBC driver / Native Protocol (Database properties)

Driver : oracle.jdbc.driver.OracleDriver

| |
pkg name driver class name

URL : jdbc:oracle:thin:@localhost:1521:XE

----- | | | |
| | | |
sub protocol hostname portno logical db name

Username : system

Password : admin

Eclipse

- IDE : JEE IDE
- Environment : Java Environment
- Flavours : Kepler,Indigo,Luna,Mars and etc.
- Vendor : Eclipse Foundation
- Website : www.eclipse.org
- Opensource : Open source
- Format : Zip format
- Download :

<https://www.eclipse.org/downloads/packages/release/kepler/sr2/eclipse-ide-java-ee-developers>

Note:

Extract the Eclipse software.

ex:

right click to eclipse software --> extract file -->
select 'E' drive --> ok.

Steps to develop first JDBC application using Eclipse IDE

step1:

Launch eclipse IDE by choosing workspace location.

step2:

Create a java project i.e IH-JAVA-024.

ex:

File --> new --> project --> Java project --> Next -->
Name : IH-JAVA-024 --> Next --> finish.

step3:

Add "ojdbc14.jar" file in project build path.

ex:

Right click to IH-JAVA-024 project --> build path -->
configuration build path --> libraries --> Add external jars
--> select ojdbc14.jar file --> open --> ok.

step4:

Create a "com.ihub.www" package inside "src" folder.

ex:

Right click to src folder --> new --> package -->
Name : com.ihub.www --> finish.

step5:

Create a JDBC application inside "com.ihub.www" package.

ex:

Right click to com.ihub.www package --> new -->
class --> Name : SelectApp --> finish.

SelectApp.java

package com.ihub.www;

```
//ctrl+shift+o
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class SelectApp
{
    public static void main(String[] args) throws Exception
    {
```

```

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
Statement st=con.createStatement();
ResultSet rs=st.executeQuery("select * from student");

while(rs.next())
{
    System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
}

rs.close();
st.close();
con.close();
}
}

```

step6:

Run the JDBC Application.

ex:

Right click to SelectApp.java --> run as --> Java Application.

Q)Write a JDBC application to select student name and student address based on student number?

ex:

```
package com.ihub.www;
```

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Scanner;
```

```
public class SelectApp2
```

```
{
```

```
    public static void main(String[] args) throws Exception
```

```

{
    Scanner sc=new Scanner(System.in);

    System.out.println("Enter the student no :");
    int no=sc.nextInt();

    Class.forName("oracle.jdbc.driver.OracleDriver");

    Connection
    con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin
");

    Statement st=con.createStatement();

    String qry="select sname,sadd from student where sno="+no;

    ResultSet rs=st.executeQuery(qry);

    int cnt=0;
    while(rs.next())
    {
        System.out.println(rs.getString(1)+" "+rs.getString(2));
        cnt=1;
    }

    if(cnt==0)
        System.out.println("No Rows Selected");

    rs.close();
    st.close();
    con.close();
}
}

```

Non-Select Queries

Q)Write a JDBC application to insert a record into student table?

```

package com.ihub.www;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.util.Scanner;
public class InsertApp {

```

```

public static void main(String[] args) throws Exception
{
    Scanner sc=new Scanner(System.in);

    System.out.println("Enter the student no :");
    int no=sc.nextInt();

    System.out.println("Enter the student name :");
    String name=sc.next();

    System.out.println("Enter the student address :");
    String add=sc.next();

    //convert inputs according to SQL query.
    name=""+name+"";
    add=""+add+"";

    Class.forName("oracle.jdbc.driver.OracleDriver");

    Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");

    Statement st=con.createStatement();

    String qry="insert into student values("+no+","+name+","+add+")";

    int result=st.executeUpdate(qry);

    if(result==0)
        System.out.println("No Record Inserted");
    else
        System.out.println(result+" Record Inserted");

    st.close();
    con.close();

}
}

Q)Write a JDBC application to update student name based on student number?

```

package com.ihub.www;

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.util.Scanner;

public class UpdateApp {

    public static void main(String[] args) throws Exception
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the student name :");
        String name=sc.next();

        System.out.println("Enter the student no :");
        int no=sc.nextInt();

        //converting inputs according to SQL query
        name=""+name+"";
        Class.forName("oracle.jdbc.driver.OracleDriver");

        Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
        Statement st=con.createStatement();

        String qry="update student set sname="+name+" where sno="+no;

        int result=st.executeUpdate(qry);

        if(result==0)
            System.out.println("No Record updated");
        else
            System.out.println(result+" Record updated");

        st.close();
        con.close();
    }
}
```

Q)Write a jdbc application to delete a student record based on student no?

```
package com.ihub.www;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.util.Scanner;

public class DeleteApp {

    public static void main(String[] args) throws Exception
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the student no :");
        int no=sc.nextInt();

        Class.forName("oracle.jdbc.driver.OracleDriver");

        Connection
        con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin
        ");

        Statement st=con.createStatement();

        String qry="delete from student where sno="+no;

        int result=st.executeUpdate(qry);

        if(result==0)
            System.out.println("No Record deleted");
        else
            System.out.println(result+" Record deleted");

        st.close();
        con.close();
    }
}
```

Q) Types of Statement objects in JDBC?

We have three Statement objects in JDBC.

1) Simple Statement

It is an object of underlying supplied java class which implements java.sql.Statement interface.

2) PreparedStatement

It is an object of underlying supplied java class which implements java.sql.PreparedStatement interface.

3) CallableStatement

It is an object of underlying supplied java class which implements java.sql.CallableStatement interface.

SQL Injection problem

Along with input values if we pass special SQL instructions which change behaviour of a query and behaviour of an application is called SQL injection problem.

Here special SQL instruction means comment in SQL i.e (--).

While dealing with simple Statement object there is a chance of raising SQL injection problem.

ex:

```
Username : raja'--  
password : hyd
```

Valid Credentials

Userlist table

```
drop table userlist;  
create table userlist(uname varchar2(10),pwd varchar2(10));  
insert into userlist values('raja','rani');  
insert into userlist values('king','kingdom');  
commit;
```

ex:

```
package com.ihub.www;
```

```
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.Statement;  
import java.util.Scanner;
```

```
public class SQLInjProbApp
{
    public static void main(String[] args) throws Exception
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the username :");
        String name=sc.next();

        System.out.println("Enter the password :");
        String pass=sc.next();

        //converting inputs according to SQL query
        name=""+name+"";
        pass=""+pass+"";

        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin
");

        Statement st=con.createStatement();

        String qry="select count(*) from userlist where uname='"+name+"' and
pwd='"+pass+';

        ResultSet rs=st.executeQuery(qry);

        int result=0;
        while(rs.next())
        {
            result=rs.getInt(1);
        }

        if(result==0)
            System.out.println("Invalid Credentials");
        else
            System.out.println("Valid Credentials");

        rs.close();
        st.close();
        con.close();
    }
}
```

Type1 JDBC Driver Architecture/JDBC-ODBC Bridge Driver (Partly java driver)

Type1 JDBC driver is not designed to interact with database software directly.

It is designed to take the support of ODBC drivers and Vendor DB libraries to locate and interact with database software.

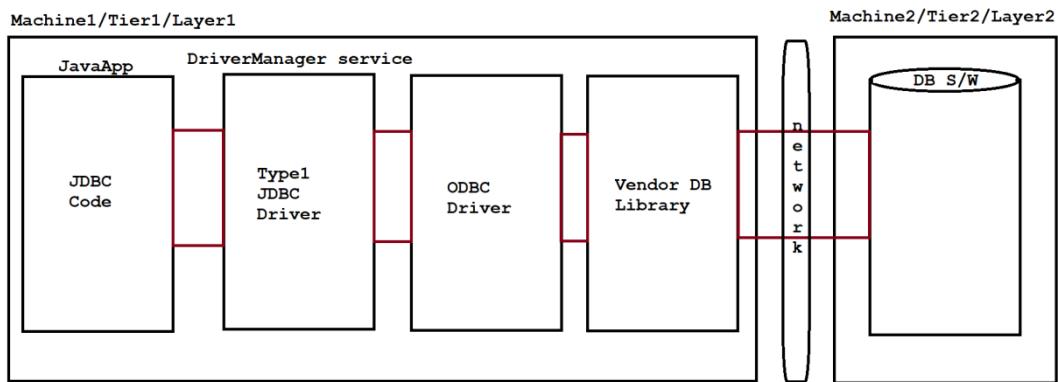


Diagram: jdbc4.1

Advantages:

- > It is a built-in driver of JDK.
- > Using Type1 JDBC driver we can interact with any database software.

Disadvantages:

- > This driver performance is low. It is not suitable for medium and large scale applications. Hence it is not a industry standard driver.
- > To work with type1 jdbc driver we need to arrange ODBC driver and vendor db library.
- > Since ODBC driver and vendor db library present at client side so it is not suitable for untrusted applets to database communication.

Type2 JDBC Driver Architecture / Native API (partly java driver)

Type2 JDBC driver is not designed to interact with database software directly.

It is designed to take the support of vendor db library to locate and interact with database softwares.

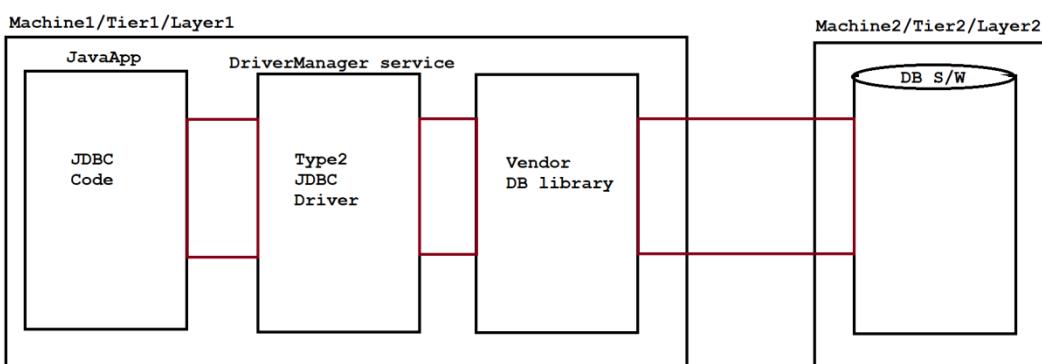


Diagram: jdbc4.2

Advantages:

- > This driver will give better performance when compare to Type1 JDBC driver.
- > It will not take the support of ODBC driver.

Disadvantages:

- > This driver performance is quit slow. It is not suitable for medium and large scale projects. Hence it is not a industry standard driver.
- > To work with Type2 JDBC driver we need to arrange vendor db library separately.
- > Since vendor db library present at client side so it is not suitable for untrusted applets to database communication.
- > For every database software we need to arrange Type2 jdbc driver separately.

Type4 JDBC driver / Native Protocol (Java driver) / thin driver

Type4 JDBC driver is not designed to take the support of ODBC driver and vendor db library.

It is designed to interact with database software directly.

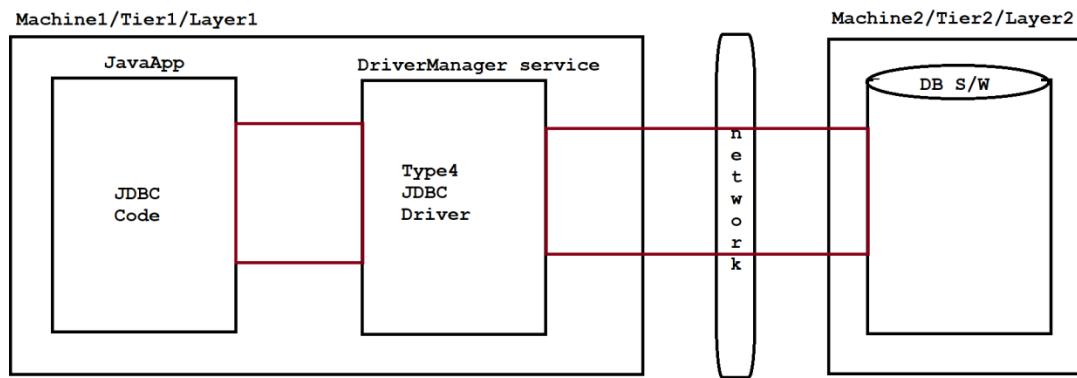


Diagram: jdbc4.3

Advantages:

- > This driver will give better performance when compare to Type 1 & 2 driver.
- > It is developed in java so it will give platform independency.
- > It will not take the support of odbc driver and vendor db library.
- > It is suitable for medium and large scale project. Hence it is a industry standard driver.
- > Since odbc driver and vendor db library not present at client side so it is suitable for untrusted applets to database communication.

Disadvantages:

- > It is a built-in driver of JDK.
- > For every database we need to arrange type4 jdbc driver separately.

JDBC Connection pool

It is a factory containing set of readily available JDBC Connection objects before actual being used.

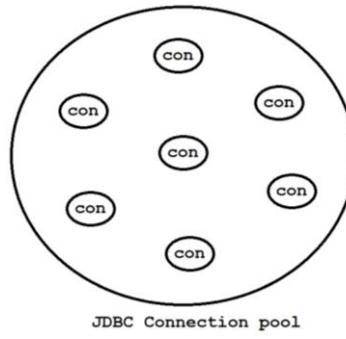


Diagram: jdbc4.4

JDBC Connection pool represent connectivity with same database software.

Advantages:

- > It will give reusable JDBC Connection objects.
- > With minimum number of Connection objects we can interact with multiple clients.
- > A user is not responsible to create, manage and destroy JDBC connection objects. A JDBC Connection pool is responsible to create, manage and destroy jdbc Connection object in JDBC Connection pool.

Type3 JDBC Driver Architecture / Net Protocol

Web server, Proxy server or IDE's server contains JDBC Connection pool representing reusable JDBC Connection objects.

Type3 JDBC driver is not designed to interact with database software directly.

Type3 JDBC driver is designed to interact with web server or proxy server to get one reusable JDBC Connection object from JDBC Connection pool.

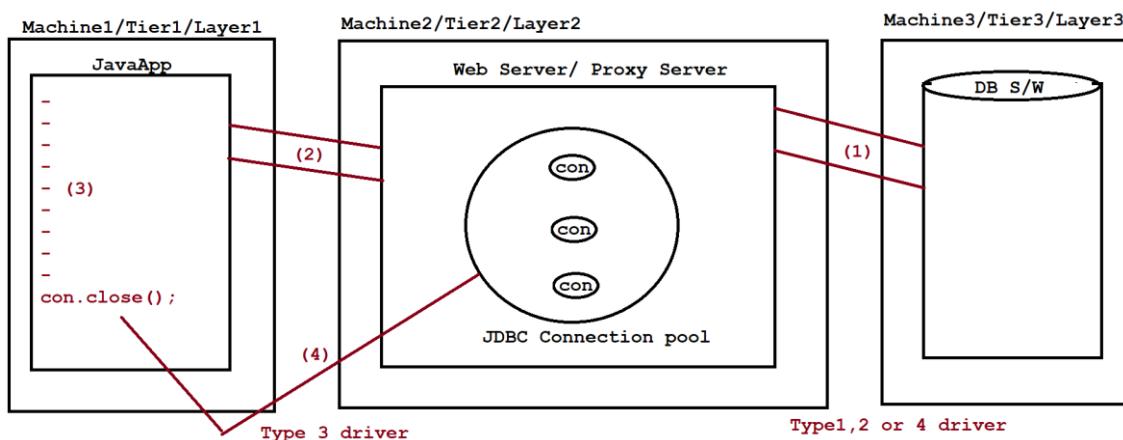


Diagram: jdbc5.1

With respect to the diagram:

- 1) Webserver or proxy server interacts with database software and gets JDBC Connection objects in JDBC Connection pool.
- 2) Java application interacts with web server or proxy server and gets one reusable JDBC Connection object from JDBC Connection pool.
- 3) Our application uses JDBC Connection object to create other connection objects.
- 4) Once if we call `con.close()` then JDBC Connection object goes back to JDBC Connection pool.

Q) How many JDBC Connection objects are there in JDBC?

We have two JDBC Connection objects.

1) Direct JDBC Connection object

A JDBC Connection object which is created by the user is called direct JDBC Connection object.

ex:

```
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
```

2) Pooled JDBC Connection object

A JDBC Connection object which is gathered from JDBC Connection pool is called pooled JDBC Connection object.

Limitations with simple Statement object

- > It is not suitable to perform same query for multiple times with same values or different values.
- > We can't use String values directly to query parameter without any conversion.
- > Framing query with variables is quite complex.
- > It raises SQL injection problem.
- > It does not allow us to insert date values in a database table column.
- > It does not allow us to insert LOB values in a database table column.

To overcome this limitations we need to use PreparedStatement object.

Pre-compiled SQL Query

Our query goes to database software without inputs and becomes parsed query either it is executed or not is called pre-compiled SQL query.

PreparedStatement object deals with pre-compiled SQL query.

Working with PreparedStatement object

step1:

Create a query with placeholders or parameters.

ex:

```
String qry="insert into student values(?, ?, ?);
```

step2:

Convert SQL query to precompiled SQL query.

ex:

```
PreparedStatement ps=con.prepareStatement(qry);
```

step3:

Set the values to query parameters.

ex:

```
ps.setInt(1,no);
```

```
ps.setString(2,name);
ps.setString(3,add);
```

step4:

Execute pre-compiled SQL Query.

ex:

```
ps.executeUpdate();
```

step5:

Close PreparedStatement object.

ex:

```
ps.close();
```

Q)Write a jdbc application to insert a record into student table using PreparedStatement object?

```
package com.ihub.www;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.util.Scanner;
public class PSInsertApp
{
    public static void main(String[] args) throws Exception
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the student no :");
        int no=sc.nextInt();
        System.out.println("Enter the student name :");
        String name=sc.next();
        System.out.println("Enter the student address :");
        String add=sc.next();

        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
        String qry="insert into student values(?, ?, ?)";
        PreparedStatement ps=con.prepareStatement(qry);
        //set the values to place holders
        ps.setInt(1,no);
        ps.setString(2,name);
        ps.setString(3,add);
        //execute
        int result=ps.executeUpdate();
        if(result==0)
```

```

        System.out.println("Record Not Inserted");
    else
        System.out.println("Record Inserted");

    ps.close();
    con.close();
}
}

```

Q)Write a jdbc application to update student name based on student number?

```

package com.ihub.www;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.util.Scanner;
public class PSUpdateApp
{
    public static void main(String[] args) throws Exception
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the student no :");
        int no=sc.nextInt();
        System.out.println("Enter the student name :");
        String name=sc.next();

        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin
");

        String qry="update student set sname=? where sno=?";
        PreparedStatement ps=con.prepareStatement(qry);

        //set the values
        ps.setString(1,name);
        ps.setInt(2,no);

        //execute
        int result=ps.executeUpdate();

        if(result==0)
            System.out.println("No Record Updated");
        else
            System.out.println("Record Updated");
    }
}

```

```
        ps.close();
        con.close();
    }
}
```

Q)Write a JDBC Application to delete a student record based on student number?

```
package com.ihub.www;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.util.Scanner;

public class PSDeleteApp
{
    public static void main(String[] args) throws Exception
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the student no :");
        int no=sc.nextInt();

        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin
");

        String qry="delete from student where sno=?";

        PreparedStatement ps=con.prepareStatement(qry);

        //set the values
        ps.setInt(1,no);

        //execute
        int result=ps.executeUpdate();

        if(result==0)
            System.out.println("No Record Deleted");
        else
            System.out.println("Record Deleted");

        ps.close();
        con.close();
    }
}
```

```
    }  
}
```

Solution for SQL Injection problem

```
package com.ihub.www;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.util.Scanner;  
  
public class SolForSQLInjProb  
{  
    public static void main(String[] args) throws Exception  
    {  
        Scanner sc=new Scanner(System.in);  
        System.out.println("Enter the username :");  
        String name=sc.next();  
        System.out.println("Enter the password :");  
        String pass=sc.next();  
  
        Class.forName("oracle.jdbc.driver.OracleDriver");  
        Connection  
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin  
");  
  
        String qry="select count(*) from userlist where uname=? and pwd=?";  
  
        PreparedStatement ps=con.prepareStatement(qry);  
  
        //set the values  
        ps.setString(1,name);  
        ps.setString(2,pass);  
  
        //execute  
        ResultSet rs=ps.executeQuery();  
  
        int result=0;  
        while(rs.next())  
        {  
            result=rs.getInt(1);  
        }  
        if(result==0)
```

```

        System.out.println("Invalid Credentials ");
    else
        System.out.println("Valid Credentials ");

    rs.close();
    ps.close();
    con.close();
}
}

```

DatabaseMetaData

DatabaseMetaData is an interface which is present in java.sql package.

DatabaseMetaData provides metadata of a database.

DatabaseMetaData gives information about database product name, database product version, database driver name, database driver version, database username and etc.

We can create DatabaseMetaData object as follow.

ex:

```
DatabaseMetaData dbmd=con.getMetaData();
```

ex:

```

package com.ihub.www;
import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
public class DBMDApp {

    public static void main(String[] args) throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection
        con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin
");
    }
}
```

```

        DatabaseMetaData dbmd=con.getMetaData();

        System.out.println(dbmd.getDatabaseProductName());
        System.out.println(dbmd.getDatabaseProductVersion());
        System.out.println(dbmd.getDriverName());
        System.out.println(dbmd.getDriverVersion());
        System.out.println(dbmd.getUserName());
        con.close();
    }

}
```

ResultSetMetaData

ResultSetMetaData is an interface which is present in java.sql package.

ResultSetMetaData provides metadata of a table.

ResultSetMetaData gives information about number of columns , name of a columns, type of columns, size of a columns and etc.

We can create ResultSetMetaData object as follow.

ex:

```
ResultSetMetaData rsmd=rs.getMetaData();
```

ex:

```
package com.ihub.www;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;

public class RSMDApp
{
    public static void main(String[] args) throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin
");
        Statement st=con.createStatement();
        String qry="select * from student";
        ResultSet rs=st.executeQuery(qry);

        ResultSetMetaData rsmd=rs.getMetaData();

        System.out.println(rsmd.getColumnCount());
        System.out.println(rsmd.getColumnName(1));
        System.out.println(rsmd.getColumnTypeName(2));
        System.out.println(rsmd getColumnDisplaySize(2));

        rs.close();
        st.close();
        con.close();
    }
}
```

Working with Date values

While dealing with DOB,DOA,DOR,DOD and etc we need to insert and retrieve date values. It is never recommended to store date values in the form of String because we can't compare two dates.

Every database software supports different date patterns.

ex:

Oracle - dd-MMM-yy

MySQL - yyyy-MM-dd

A `java.util.Date` class object is not suitable to perform database operation.

A `java.sql.Date` class object is suitable to perform database operation.

Using simple Statement object we can place date values to query parameters.

To overcome this limitation we need to use PreparedStatement object.

Once JDBC driver will get the date value then it will insert in the pattern which is supported by underlying database software.

Standard procedure to insert date

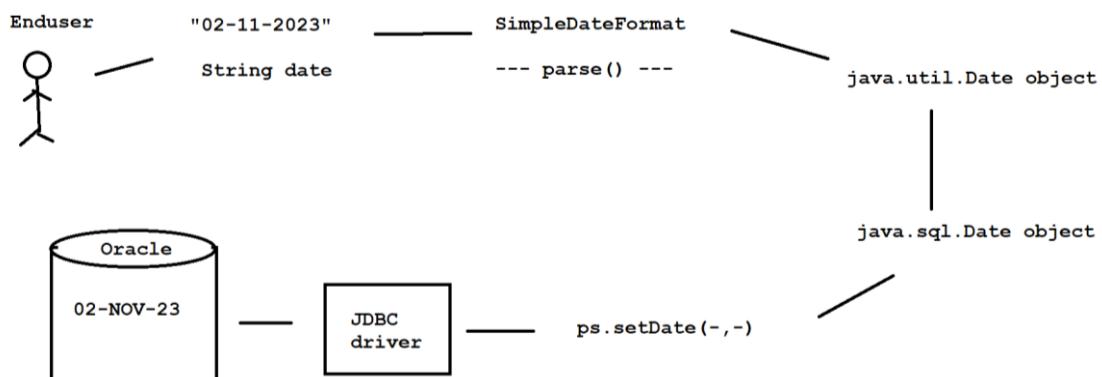


Diagram: jdbc6.1

With the respect to the diagram:

- 1) Enduser will pass date value in the form of String.
- 2) A `parse()` method of `java.text.SimpleDateFormat` class converts String date to `java.util.Date` class object.
- 3) Our application converts `java.util.Date` class object to `java.sql.Date` class object.
- 4) A `ps.setDate(-,-)` method is used to set date value to query parameter.
- 5) Once JDBC driver gets date value then it will insert in the pattern which is supported by underlying database software.

emp1 table

```
drop table emp1;  
create table emp1(eid number(3),ename varchar2(10),edoj date);
```

DateInsertApp.java

```
package com.ihub.www;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.text.SimpleDateFormat;
import java.util.Scanner;

public class DateInsertApp {

    public static void main(String[] args) throws Exception
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the employee id ");
        int id=sc.nextInt();

        System.out.println("Enter the employee name :");
        String name=sc.next();

        System.out.println("Enter the employee DOJ(dd-MM-yyyy) :");
        String sdoj=sc.next();

        //converting string date to java.util.Date class object
        SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
        java.util.Date udoj=sdf.parse(sdoj);

        //converting util date object to sql date object
        long ms=udoj.getTime();
        java.sql.Date sqldoj=new java.sql.Date(ms);

        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin
");

        String qry="insert into emp1 values(?, ?, ?)";

        PreparedStatement ps=con.prepareStatement(qry);

        //set the values
        ps.setInt(1,id);
        ps.setString(2,name);
```

```

        ps.setDate(3,sqldoj);

        //execute
        int result=ps.executeUpdate();

        if(result==0)
            System.out.println("No Record inserted");
        else
            System.out.println("Record inserted");

        ps.close();
        con.close();

    }

}

```

DateRetrieveApp.java

```

package com.ihub.www;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.text.SimpleDateFormat;

public class DateRetrieveApp
{
    public static void main(String[] args) throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin
");
        Statement st=con.createStatement();
        String qry="select * from emp1";
        ResultSet rs=st.executeQuery(qry);
        while(rs.next())
        {
            int id=rs.getInt(1);
            String name=rs.getString(2);
            java.sql.Date sqldoj=rs.getDate(3);

            //converting sql date to util date
        }
    }
}

```

```

        java.util.Date udoj=(java.util.Date)sqldoj;

        SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
        String sdoj=sdf.format(udoj);

        System.out.println(id+" "+name+" "+sdoj);
    }

    rs.close();
    st.close();
    con.close();

}

}

```

Working with LOB values

Files are known as LOB's.

We have two types of LOB's.

1) BLOB (Binary Large Object)

ex:

images, audio, video, avi file and etc.

2) CLOB (Character Large Object)

ex:

text, doc file , advanced text file and etc.

While dealing with matrimonial applications, job portal applications, profile management applications and etc. we need to insert and retrieve LOB values.

Using simple Statement object we can set LOB values directly to query parameters. We need to take the support of PreparedStatement object.

We can set LOB values to query parameter by using following methods.

ex:

```

ps.setBinaryStream(-,-,-) / ps.setBLOB(-,-,-)
ps.setCharacterStream(-,-,-) / ps.setCLOB(-,-,-)

```

emp2 table

=====

drop table emp2;

create table emp2(eid number(3),ename varchar2(10), ephoto BLOB);

PhotoInsertApp.java

```
package com.ihub.www;

import java.io.File;
import java.io.FileInputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.util.Scanner;

public class PhotoInsertApp {

    public static void main(String[] args) throws Exception
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the employee id :");
        int id=sc.nextInt();

        System.out.println("Enter the employee name :");
        String name=sc.next();

        //locate the file
        File f=new File("src/com/ihub/www/rock.jpeg");
        FileInputStream fis=new FileInputStream(f);

        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection
        con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin
");

        String qry="insert into emp2 values(?, ?, ?)";

        PreparedStatement ps=con.prepareStatement(qry);

        //set the values
        ps.setInt(1,id);
        ps.setString(2,name);
        ps.setBinaryStream(3,fis,(int)f.length());

        //execute
        int result=ps.executeUpdate();
        if(result==0)
            System.out.println("No Record inserted");
        else
    }
}
```

```
        System.out.println("Record inserted");

        ps.close();
        con.close();

    }

}
```

PhotoRetrieveApp.java

```
package com.ihub.www;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class PhotoRetrieveApp
{
    public static void main(String[] args) throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
        Statement st=con.createStatement();

        String qry="select * from emp2";

        ResultSet rs=st.executeQuery(qry);

        while(rs.next())
        {
            InputStream is=rs.getBinaryStream(3);

            FileOutputStream fos=new FileOutputStream("E:\\IHUB-Training-Batches\\IH-JAVA-024\\praveen.png");

            int byteReads=0;
            byte[] buff=new byte[255];
```

```

        while((byteReads=is.read(buff))!=-1)
        {
            fos.write(buff, 0, byteReads);
        }

        fos.close();
    }

    System.out.println("Please check the location");
    rs.close();
    st.close();
    con.close();

}
}

```

JDBC Flexible Application

In JDBC , Connection object consider as heavy weight object.

It is never recommended to create JDBC Connection object in every JDBC application.

We need to a create a class which returns JDBC Connection object.

DBConnection.java

```

package com.ihub.www;
import java.sql.Connection;
import java.sql.DriverManager;
public class DBConnection
{
    static Connection con=null;
    public static Connection getConnection()
    {
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system",
"admin");
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }

        return con;
    }
}

```

FlexibleApp.java

```
package com.ihub.www;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;

public class FlexibleApp
{
    public static void main(String[] args) throws Exception
    {
        Connection con=DBCConnection.getConnection();

        Statement st=con.createStatement();

        String qry="select * from student";

        ResultSet rs=st.executeQuery(qry);

        while(rs.next())
        {
            System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
        }

        rs.close();
        st.close();
        con.close();
    }
}
```

Working with Properties file

In regular intervals, Our DBA will change username and password for security reasons.

It is never recommended to pass database properties directly to the application.

It is always recommended to read database properties from properties file.

A properties file contains key and value pair.

dbdetails.properties

```
driver=oracle.jdbc.driver.OracleDriver
url=jdbc:oracle:thin:@localhost:1521:XE
username=system
password=admin
```

PropertiesApp.java

```
package com.ihub.www;

import java.io.FileInputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Properties;
public class PropertiesApp
{
    public static void main(String[] args) throws Exception
    {
        //locate properties file
        FileInputStream fis=new
FileInputStream("src/com/ihub/www/dbdetails.properties");

        //create Properties class object
        Properties p=new Properties();

        //reading the data from file to class
        p.load(fis);

        //reading the data from class
        String s1=p.getProperty("driver");
        String s2=p.getProperty("url");
        String s3=p.getProperty("username");
        String s4=p.getProperty("password");

        Class.forName(s1);
        Connection con=DriverManager.getConnection(s2,s3,s4);
        Statement st=con.createStatement();
        String qry="select * from student";
        ResultSet rs=st.executeQuery(qry);
        while(rs.next())
        {
            System.out.println(rs.getInt(1)+"          "+rs.getString(2)+""
"+rs.getString(3));
        }
        rs.close();
        st.close();
        con.close();
    }
}
```

Thin-Client/Fat-Server Application

Every JDBC application consider as Thin-Client/Fat-Server application.

To develop thin-client/fat-server application , we need to save our business logic and persistence logic in database software in the form of PL/SQL procedures and Functions.

To deal with PL/SQL procedures and functions we need to use CallableStatement object.

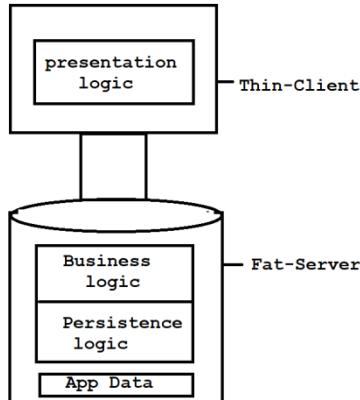


Diagram: jdbc7.1

PL/SQL procedure

```
create or replace procedure first_proc(A IN number,B IN number,C OUT number)
```

```
IS
```

```
BEGIN
```

```
C:=A+B;
```

```
END;
```

```
/
```

```
package com.ihub.www;
```

```
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Types;
```

```
public class CallableStmtApp
```

```
{
```

```
    public static void main(String[] args) throws Exception
    {

```

```
        Class.forName("oracle.jdbc.driver.OracleDriver");

```

```
        Connection

```

```
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin
");
```

```
CallableStatement cst=con.prepareCall("{CALL first_proc(?, ?, ?)}");
```

```
//register OUT parameter
```

```

cst.registerOutParameter(3, Types.INTEGER);

//set the values to IN parameter
cst.setInt(1, 10);
cst.setInt(2, 20);

//execute
cst.execute();

//gather the result
int result=cst.getInt(3);
System.out.println("sum of two numbers is =" +result);

cst.close();
con.close();
}

}

```

Types of ResultSet Objects

We have two types of ResultSet objects.

- 1)Non-Scrollable ResultSet object
- 2)Scrollable ResultSet object

1) Non-Scrollable ResultSet object

A ResultSet object which allows us to read the records sequentially, unidirectionally is called non-scrollable ResultSet object.

Bydefault every ResultSet object is a non-scrollable ResultSet object.

If JDBC Statement object is created without type,mode value then ResultSet object is called Non-scrollable ResultSet object.

ex:

```

Statement st=con.createStatement();
ResultSet rs=st.executeQuery("select * from student");

```

2) Scrollable ResultSet object

A ResultSet object which allows us to read the records non-sequentially, bidirectionally, randomly is called scrollable ResultSet object.

If JDBC Statement object is created with type,mode value then ResultSet object is called Non-scrollable ResultSet object.

ex:

```

Statement st=con.createStatement(type,mode);
ResultSet rs=st.executeQuery("select * from student");

```

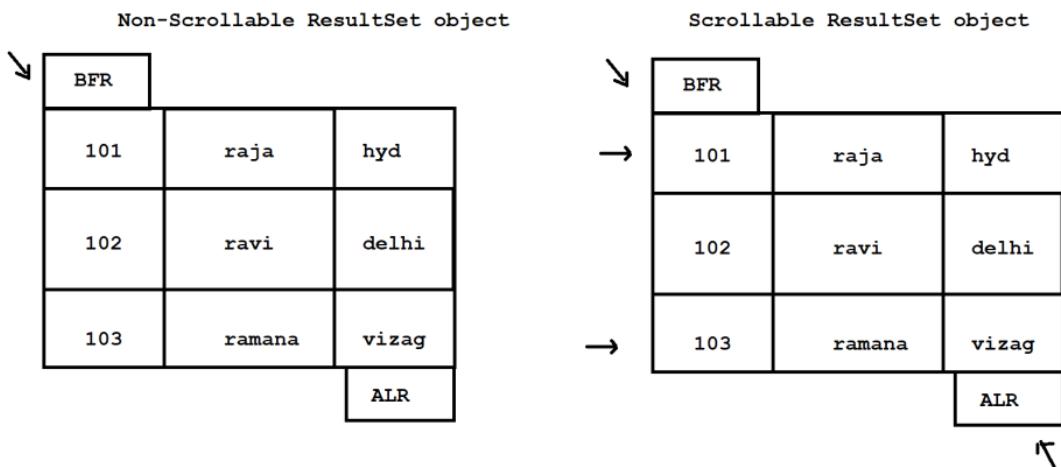


Diagram: jdbc8.1

We have two type values.

ex:

`ResultSet.TYPE_SCROLL_SENSITIVE`
`ResultSet.TYPE_SCROLL_INSENSITIVE`

We have two mode values.

ex:

`ResultSet.CONCUR_READ_ONLY`
`ResultSet.CONCUR_UPDATABLE`

Various methods present in Scrollable ResultSet object

`rs.next()`

It will move the record pointer to next position.

`rs.getRow()`

It will return position of record pointer.

`rs.getXXX()`

It will return the values from record ResultSet.

`rs.close()`

It is used to close the ResultSet object.

`rs.previous()`

It will move the record pointer to previous position.

`rs.first()`

It will set the record pointer to first record.

`rs.isFirst()`

It is used to check record pointer is in first position or not.

`rs.last()`

It will set the record pointer to last record.

`rs.isLast()`

It is used to check record pointer is in last position or not.

rs.beforeFirst()

It will set the record pointer to BFR position.

rs.afterLast()

It will set the record pointer to ALR position.

rs.relative(+/-)

It will move the record pointer to next position based on current position.

rs.absolute(+/-)

It will move the record pointer to next position based on BFR and ALR.

ex:

```
package com.ihub.www;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
```

```
public class ScrollableResultSetApp {
```

```
    public static void main(String[] args) throws Exception
    {
```

```
        Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
        Connection
```

```
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
");
```

```
        Statement st=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
```

```
ResultSet.CONCUR_READ_ONLY);
```

```
        String qry="select * from student";
```

```
        ResultSet rs=st.executeQuery(qry);
```

```
        //top to bottom
```

```
        while(rs.next())
```

```
{
```

```
            System.out.println(rs.getRow()+" "+rs.getInt(1)+" "+rs.getString(2)+""
"+rs.getString(3));
        }
```

```
        rs.afterLast();
```

```

        while(rs.previous())
        {
            System.out.println(rs.getRow()+" "+rs.getInt(1)+" "+rs.getString(2)+""
"+rs.getString(3));
        }

        rs.first();
        System.out.println(rs.isFirst());
        System.out.println(rs.getRow()+"      "+rs.getInt(1)+"      "+rs.getString(2)+""
"+rs.getString(3));

        rs.last();
        System.out.println(rs.isLast());//true
        System.out.println(rs.getRow()+"      "+rs.getInt(1)+"      "+rs.getString(2)+""
"+rs.getString(3));

        //rs.relative(-2);
        rs.absolute(-2);
        System.out.println(rs.getRow()+"      "+rs.getInt(1)+"      "+rs.getString(2)+""
"+rs.getString(3));

        rs.close();
        st.close();
        con.close();

    }

}

```

Batch Processing

Batch processing is used to declare multiple queries in the application and makes a single call to the database.

Each query we need to add in a batch.

To add the query in a batch we need to use addBatch() method Statement object.

ex:

```
st.addBatch("select * from student");
```

To execute the batch we need to use executeBatch() method of Statement object.

ex:

```
int[] result=st.executeBatch();
```

ex:

```

package com.ihub.www;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class BatchProcessing
{
    public static void main(String[] args) throws Exception
    {

        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
        Statement st=con.createStatement();

        String qry1="insert into student values(104,'ramulu','pune')";
        String qry2="delete from student where sno=103";
        String qry3="update student set sname='rani' where sno=101";

        //add the queries to batch
        st.addBatch(qry1);
        st.addBatch(qry2);
        st.addBatch(qry3);

        //execute the batch
        int[] result=st.executeBatch();

        //for each loop
        int sum=0;
        for(int i:result)
        {
            sum+=i;
        }
        System.out.println("No of records effected are =" +sum);

        st.close();
        con.close();
    }
}

```

Transaction Management

Transaction represent single unit of work.

JDBC Connection interface methods we can manage the transaction management.

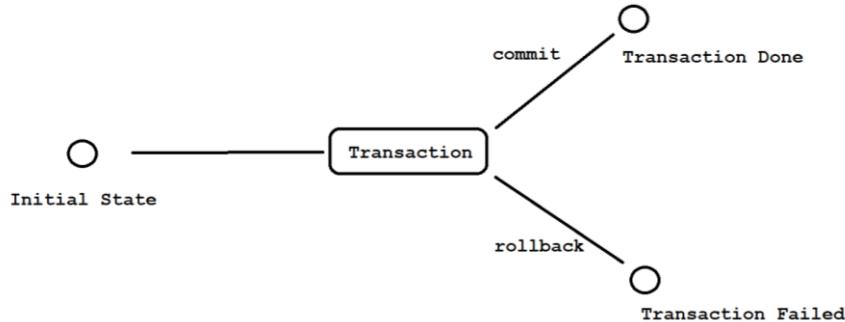


Diagram: jdbc9.1

sbi bank

```
drop table sbi;
create table sbi(accno number(6),accholder varchar2(10), accbal number(10));
insert into sbi values(111111,'sandeep',10000);
insert into sbi values(222222,'pradeep',12000);
commit;
```

kotak bank

```
drop table kotak;
create table kotak(accno number(6),accholder varchar2(10), accbal number(10));
insert into kotak values(100001,'thrishul',90000);
insert into kotak values(200002,'raju',80000);
commit;
```

```
package com.ihub.www;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.util.Scanner;
public class TXNManagementApp
{
    public static void main(String[] args) throws Exception
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the source account no :");
        int sno=sc.nextInt();
        System.out.println("Enter the destination account no :");
        int dno=sc.nextInt();
        System.out.println("Enter the amount to transfer :");
        int amt=sc.nextInt();
        Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
Connection  
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin  
");  
  
//set auto commit false  
con.setAutoCommit(false);  
  
Statement st=con.createStatement();  
  
//create the queries  
String qry1="update kotak set accbal=accbal-"+amt+" where accno="+sno;  
String qry2="update sbi set accbal=accbal-"+amt+" where accno="+dno;  
  
//add the queries to batch  
st.addBatch(qry1);  
st.addBatch(qry2);  
  
//execute the batch  
int[] result=st.executeBatch();  
  
boolean flag=true;  
for(int i:result)  
{  
    if(i==0)  
    {  
        flag=false;  
        break;  
    }  
}  
if(flag==true)  
{  
    System.out.println("Transaction Done Successfully");  
    con.commit();  
}  
else  
{  
    System.out.println("Transaction Failed!!");  
    con.rollback();  
}  
  
st.close();  
con.close();  
}  
}
```

Standard procedure to develop JDBC application

```
package com.ihub.www;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
public class StandardApp
{
    public static void main(String[] args)
    {
        final String DRIVER="oracle.jdbc.driver.OracleDriver";
        final String URL="jdbc:oracle:thin:@localhost:1521:XE";
        final String USERNAME="system";
        final String PASSWORD="admin";

        Connection con=null;
        Statement st=null;
        ResultSet rs=null;
        String qry=null;
        try
        {
            Class.forName(DRIVER);

            con=DriverManager.getConnection(URL,USERNAME,PASSWORD);
            st=con.createStatement();
            qry="select * from student";
            rs=st.executeQuery(qry);
            while(rs.next())
            {
                System.out.println(rs.getRow()+"          "+rs.getInt(1)+""
"+rs.getString(2)+" "+rs.getString(3));
            }

            rs.close();
            st.close();
            con.close();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Steps to interact with MYSQL Database

step1:

Download and Install MY/SQL Database successfully.

ex:

https://drive.google.com/file/d/1QQjWTJ9v8xz0nfuSGva1_QQwO6KDf9_c/view?usp=sharing

step2:

Connect with mysql by using password.

ex:

username : root(default)

password: root

step3:

create a SCHEMA in MYSQL.

ex:

create schema IH_JAVA_024

step4:

To check list of databases /schemas present in mysql db.

ex:

show databases;

step5:

Use IH_JAVA_024 scheme/database.

ex:

use IH_JAVA_024;

step6:

create a student table and insert the records.

ex:

```
create table student(sno int(3),sname varchar(10),sadd varchar(10));
insert into student values(101,'raja','hyd');
insert into student values(102,'raju','delhi');
insert into student values(103,'ravi','pune');
commit;
```

step7:

create a JDBC Application to select student records.

MySQLApp.java

```
package com.ihub.www;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class MySQLApp
{
    public static void main(String[] args)
    {
        final String DRIVER="com.mysql.jdbc.Driver";
        final String URL="jdbc:mysql://localhost:3306/IH_JAVA_024?characterEncoding=utf8";
        final String USERNAME="root";
        final String PASSWORD="root";
        final String QUERY="select * from student";

        Connection con=null;
        Statement st=null;
        ResultSet rs=null;
        try
        {
            Class.forName(DRIVER);

            con=DriverManager.getConnection(URL,USERNAME,PASSWORD);
            st=con.createStatement();
            rs=st.executeQuery(QUERY);
            while(rs.next())
            {
                System.out.println(rs.getRow()+" "+rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
            }

            rs.close();
            st.close();
            con.close();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

step8:

Add "mysql-connector.jar" file in project build path for mysql database.

right click to project --> built path --> configuration build path --> libraries
--> add external jars --> select mysql-connector.jar file --> open.

jar file download :

<http://www.java2s.com/Code/Jar/m/Downloadmysqlconnectorjavajar.htm>

<http://www.java2s.com/Code/Jar/m/Downloadmysqlconnectorjar.htm>

Note:

ojdbc14.jar - - for oracle

mysql-connector.jar --> for mysql

or

mysql-connector-java.jar --> for mysql

step9:

Run the jdbc application.

SERVLETS

Web Application

A web application is a collection of web resource programs having the capability to generate web pages.

We have two types of web pages.

1) static web page / passive web page

If a content not change for a web page is called static web page.

ex:

- facebook login page
- home page
- contactus page
- services page
- and etc

2) Dynamic web page / active web page

If a content change for a web page is called dynamic web page.

ex:

- live cricket score page
- stock market share value page
- gmail inbox page
- and etc

We have two types of web resource programs.

1) Static web resource program

It is used to generate static web pages.

ex:

- html program
- css program
- bootstrap program
- reactjs program
- angularjs program
- and etc.

2) Dynamic web resource program

It is used to generate dynamic web pages.

ex:

- servlet program
- jsp program
- and etc.

Based on the position and execution these web resource programs are divided into two types.

1) Client side web resource program

A web resource program which executes at client side(browser window) is called client side web resource program.

All static web resource programs are called client side web resource program.

2) Server side web resource program

A web resource program which executes at server side is called server side web resource program.

All dynamic web resource programs are called server side web resource program.

Web application and web resource program execution

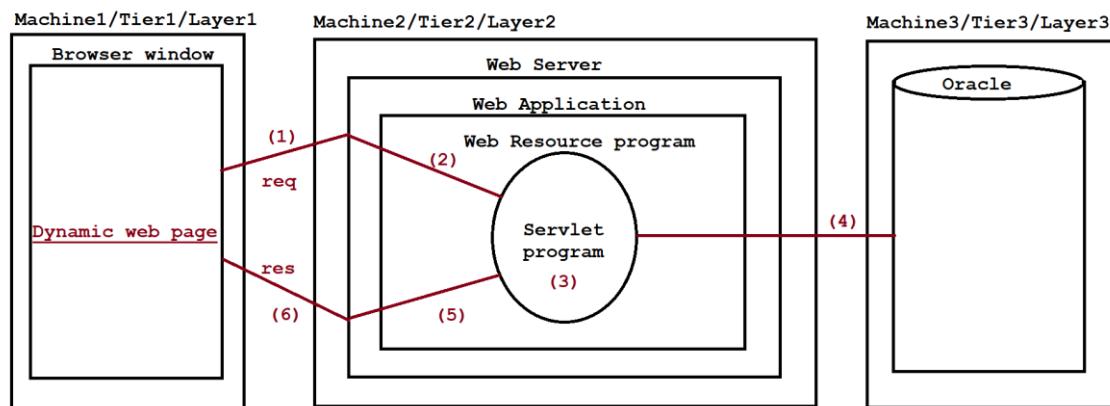


Diagram: servlet1.1

Java application will execute manually.

Web application and web resource program will execute at the time when they have requested. So there is no way to execute them manually.

With respect to the diagram:

- 1) End user will give the request to web resource program.
- 2) Web server will trap that request and passes that request to appropriate web resource program.
- 3) Web resource program will execute the logic to process the request.
- 4) Web resource program will communicate with database software if necessary.
- 5) Web resource program sends the output to web server.
- 6) Web server will give output to browser window as dynamic response.

Note:

The process of keeping the web application in a server is called deployment and reverse is called undeployment.

Web Server

A web server is a piece of software which is used to automate whole process of web application and web resource program execution.

ex:

Tomcat , Resin and etc.

Responsibilities of web server

- > It takes continues request from client.
- > It passes the request to appropriate web resource program.
- > It provides environment to deploy and undeploy the web applications.
- > It will add middleware services only to deployed web applications.
- > It provides environment to execute client side web resource programs at browser window.
- > It is used to automate whole process of web application and web resource program executions.
- > Web server will send the output to browser window as dynamic web page.

Web Container

It is a software application or a program which is use manage whole life cycle of web resource program from birth to death.

Servlet container manage whole life cycle of servlet program.

JSP container manage whole life cycle of jsp program.

some part of industry consider servlet container and jsp container are web containers.

Every server is designed to support servlet container and jsp container so we don't need to arrange them seperately.

Tomcat

=====

version	:	7.x
Creator	:	James Duncan Davidson
Vendor	:	Apache software foundation
Port No	:	8080
website	:	www.apache.org
servlet container	:	Catalina
Jsp container	:	Jasper
Download	:	

https://drive.google.com/file/d/0B9rC21sL6v0tZFdVcmxZUDA0Tms/view?usp=drive_link&resourcekey=0-VXIB_IpeWqDWwdbr1baCyA

Tomcat install will ask following things.

- 1) Http Connector Port No
- 2) Adminstrator username and password
- 3) JRE location (parallel to JDK)
- 4) Tomcat Installation location

Tomcat is a not a container.It is a server containing servlet container and jsp container.

Before 6.x version tomcat is known as web server.From 6.x version onwards tomcat is also known as application server.

Installation of tomcat server

Right click to tomcat software --> yes ---> Next --> I Agree --> select Full -->
---> Next --->

Http connector port : 2525

adminstrative username : admin

password : admin

--> next --> Next --> Install.

Keeping Tomcat server from automatic mode to manual mode

services (view local services) --> select Apache Tomcat --> click to stop link -->
double click to Apache tomcat --> startup type: manual -->Apply -->ok.

Servlet

It is a dynamic web resource program which enhanced the functionality of web server or application server.

It is a java based dynamic web resource program which is used to create dynamic web pages.

It is a single instance multithread java based web resource program which is used to develop web applications.

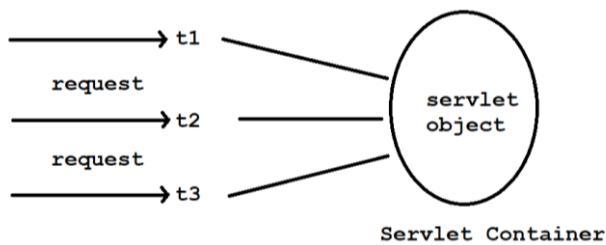


Diagram: servlet2.1

Servlet API

API is a collection of packges.

ex:

```
javax.servlet.*;  
javax.servlet.http.*;
```

Important terminology

We have following important terminology.

- 1) javax.servlet.Servlet(I)
- 2) javax.servlet.GenericServlet(AC)
- 3) javax.servlet.http.HttpServlet(C)

First web application develop having servlet program as web resource program

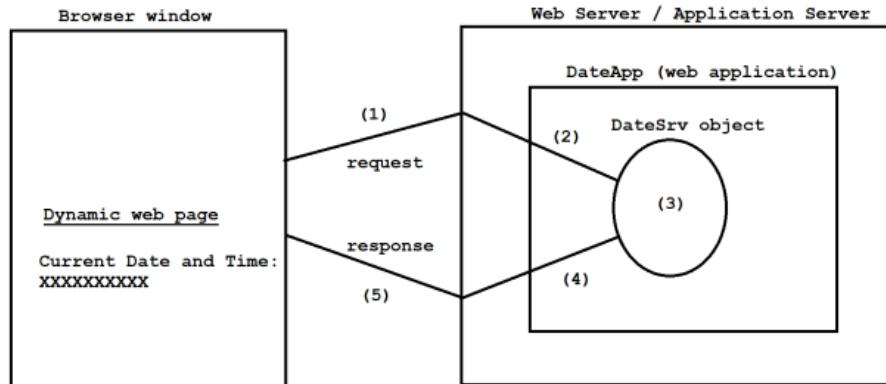


Diagram: servlet2.2

Deployment Directory structure

```
DateApp
|---Java Resources
|   |---src
|   |   |---com.ihub.www
|   |   |   |---DateSrv.java
|---Web Content
|   |---WEB-INF
|   |   |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

step1:

Launch eclipse IDE by choosing workspace location.

step2:

Create a dynamic web project i.e DateApp.

ex:

File --> new --> dynamic web project -->

project Name : DateApp

Dynamic web module version : 3.0 --> next --> next -->

select generate web.xml file(checkbox) --> finish.

step3:

Add "servlet-api.jar" in project build path.

ex:

right click to DateApp project --> build path --> configuration
built path --> libraries --> Add external jars --> select
servlet-api.jar file --> open -->ok .

step4:

Create a "com.ihub.www" package inside "src" folder.

ex:

Right click to src folder --> new --> package -->
Name : com.ihub.www --> finish.

step5:

Create a "DateSrv.java" file inside "com.ihub.www" package.

ex:

right click to com.ihub.www pkg --> new --> class -->
Name : DateSrv --> finish.

DateSrv.java

```
package com.ihub.www;

//ctrl+shift+o
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;
import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class DateSrv extends GenericServlet
{
    public void service(ServletRequest req, ServletResponse res) throws
ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        Date d = new Date();

        pw.println("<center><h1>Current Date And Time <br>" + d +
"</h1></center>");
        pw.close();
    }
}
```

step6:

Configure each servlet program in web.xml file.

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <servlet>
    <servlet-name>DateSrv</servlet-name>
    <servlet-class>com.ihub.www.DateSrv</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>DateSrv</servlet-name>
    <url-pattern>/test</url-pattern>
  </servlet-mapping>

</web-app>
```

step7:

Add "Tomcat" server to eclipse IDE.

ex:

Window --> Preferences --> servers --> runtime Environments
--> click to Add button --> select Apache --> Apache Tomcat 7
---> Next --> select Tomcat installation directory --> finish
--> ok.

step8:

Run the dynamic project i.e DateApp.

ex:

Right click to DateApp --> run as --> run on server -->
select apache tomcat 7 --> finish.

step9:

Test the application by using below request url.

ex:

url pattern
|
http://localhost:2525/DateApp/test
| | |
hostname portno webapplication

Note:

If there is a problem with web.xml file then we will get 404 Error.

If there is a problem with servlet program then we will get 500 Error.

Types of url patterns

Every servlet program will be recognized with the help of URL pattern only.

URL pattern will hide servlet name and technology name from the outsider for security reason.

Our servlet container, client, other web resource programs will recognize servlet program with url pattern.

We have three types of URL patterns.

- 1) Exact Match url pattern
- 2) Directory Match url pattern
- 3) Extension Match url pattern

Every server is designed to support these three types of url patterns.

1) Exact Match url pattern

It will start with '/' forward slash followed by a name.

ex:

web.xml

```
<url-pattern>/test</url-pattern>
```

Request url

http://localhost:2525/DateApp/test (valid)

http://localhost:2525/DateApp/best (invalid)

http://localhost:2525/DateApp/a/test (invalid)

2) Directory Match url pattern

It will start with '/' forward slash and ends with '*' symbol.

ex:

web.xml

```
<url-pattern>/x/y/*</url-pattern>
```

Request url

http://localhost:2525/DateApp/test (invalid)

http://localhost:2525/DateApp/x/y/z (valid)

http://localhost:2525/DateApp/x/y/z/test (valid)

http://localhost:2525/DateApp/y/x/z (invalid)

3) Extension Match url pattern

It will start with '*' symbol followed by an extension.

ex:

web.xml

```
<url-pattern>*.do</url-pattern>
```

Request url

http://localhost:2525/DateApp/test (invalid)

http://localhost:2525/DateApp/test.do (valid)

<http://localhost:2525/DateApp/x/y/z/test> (invalid)

<http://localhost:2525/DateApp/y/x/z.do> (valid)

MIME Types

MIME stands for Multipurpose Internet Mail Extension.

MIME describes in how many formats we can display our output in servlet.

We have following formats to display the output in servlet.

1) text/html

It is used to display the output in html format.

2) text/xml

It is used to display the output in xml format.

3) application/ms-word

It is used to display the output in word format.

4) application/vnd.ms-excel

It is used to display the output in excel format.

Deployment Directory Structure

MIMEApp

|

|---Java Resources

|

|-----src

|

|---com.ihub.www

|

|---TestSrv1.java

|---TestSrv2.java

|---TestSrv3.java

|---TestSrv4.java

|

|---Web Content

|

|---WEB-INF

|

|---web.xml

Note:

In above application we need to add "servlet-api.jar" file in project build path.

If web application contains four servlets then we need to configure each servlet program in web.xml file with multiple <servlet> and <servlet-mapping> tags.

TestSrv1.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class TestSrv1 extends GenericServlet
{
    public void service(ServletRequest req, ServletResponse res) throws
    ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        pw.println("<table border='1'>");
        pw.println("<tr><th>SNO</th><th>SNAME</th><th>SADD</th></tr>");
        pw.println("<tr><td>101</td><td>Alan</td><td>USA</td></tr>");
        pw.println("<tr><td>102</td><td>Jose</td><td>UAE</td></tr>");
        pw.println("<tr><td>103</td><td>Nelson</td><td>UK</td></tr>");
        pw.println("</table>");

        pw.close();
    }
}
```

TestSrv2.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class TestSrv2 extends GenericServlet
{
```

```

    public void service(ServletRequest req,ServletResponse res) throws
ServletException,IOException
    {
        PrintWriter pw=res.getWriter();
        res.setContentType("text/xml");

        pw.println("<table border='1'>");
        pw.println("<tr><th>SNO</th><th>SNAME</th><th>SADD</th></tr>");
        pw.println("<tr><td>101</td><td>Alan</td><td>USA</td></tr>");
        pw.println("<tr><td>102</td><td>Jose</td><td>UAE</td></tr>");
        pw.println("<tr><td>103</td><td>Nelson</td><td>UK</td></tr>");
        pw.println("</table>");

        pw.close();
    }
}

```

TestSrv3.java

```

package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class TestSrv3 extends GenericServlet
{
    public void service(ServletRequest req,ServletResponse res) throws
ServletException,IOException
    {
        PrintWriter pw=res.getWriter();
        res.setContentType("application/ms-word");

        pw.println("<table border='1'>");
        pw.println("<tr><th>SNO</th><th>SNAME</th><th>SADD</th></tr>");
        pw.println("<tr><td>101</td><td>Alan</td><td>USA</td></tr>");
        pw.println("<tr><td>102</td><td>Jose</td><td>UAE</td></tr>");
        pw.println("<tr><td>103</td><td>Nelson</td><td>UK</td></tr>");
        pw.println("</table>");

        pw.close();
    }
}

```

```
    }  
}
```

TestSrv4.java

```
package com.ihub.www;  
  
import java.io.IOException;  
import java.io.PrintWriter;  
  
import javax.servlet.GenericServlet;  
import javax.servlet.ServletException;  
import javax.servlet.ServletRequest;  
import javax.servlet.ServletResponse;  
  
public class TestSrv4 extends GenericServlet  
{  
    public void service(ServletRequest req, ServletResponse res) throws  
    ServletException, IOException  
    {  
        PrintWriter pw = res.getWriter();  
        res.setContentType("application/vnd.ms-excel");  
  
        pw.println("<table border='1'>");  
        pw.println("<tr><th>SNO</th><th>SNAME</th><th>SADD</th></tr>");  
        pw.println("<tr><td>101</td><td>Alan</td><td>USA</td></tr>");  
        pw.println("<tr><td>102</td><td>Jose</td><td>UAE</td></tr>");  
        pw.println("<tr><td>103</td><td>Nelson</td><td>UK</td></tr>");  
        pw.println("</table>");  
  
        pw.close();  
    }  
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
         xmlns="http://java.sun.com/xml/ns/javaee"  
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
                           http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">  
  
<servlet>  
    <servlet-name>TestSrv1</servlet-name>  
    <servlet-class>com.ihub.www.TestSrv1</servlet-class>  
</servlet>
```

```

<servlet-mapping>
    <servlet-name>TestSrv1</servlet-name>
    <url-pattern>/html</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>TestSrv2</servlet-name>
    <servlet-class>com.ihub.www.TestSrv2</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>TestSrv2</servlet-name>
    <url-pattern>/xml</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>TestSrv3</servlet-name>
    <servlet-class>com.ihub.www.TestSrv3</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>TestSrv3</servlet-name>
    <url-pattern>/word</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>TestSrv4</servlet-name>
    <servlet-class>com.ihub.www.TestSrv4</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>TestSrv4</servlet-name>
    <url-pattern>/excel</url-pattern>
</servlet-mapping>

</web-app>

```

request url

<http://localhost:2525/MIMEApp/html>
<http://localhost:2525/MIMEApp/xml>
<http://localhost:2525/MIMEApp/word>
<http://localhost:2525/MIMEApp/excel>

Types of communication

We can communicate to servlet program in three ways.

- 1) Browser to servlet communication
- 2) HTML to servlet communication
- 3) Servlet to servlet communication

In browser to servlet communication we need to type our request url in browser address bar. But typing request url in browser address is quite complex.

To overcome this limitation we need to use HTML to servlet communication.

In html to servlet communication we can forward the request to servlet program using html based hyperlinks and form pages.

A request which is generated by using hyperlink does not carry the data.

But a request which is generated by using form page will carry the data.

In html based hyperlink to servlet communication we need to type our request url as href url.

ex:

```
<a href="http://localhost:2525/DateApp/test"> click Here </a>
```

In html based form page to servlet communication we need to type our request url as action url.

ex:

```
<form action="http://localhost:2525/DateApp/test">
  -
  -
</form>
```

Example application on Html based hyperlink to servlet communication

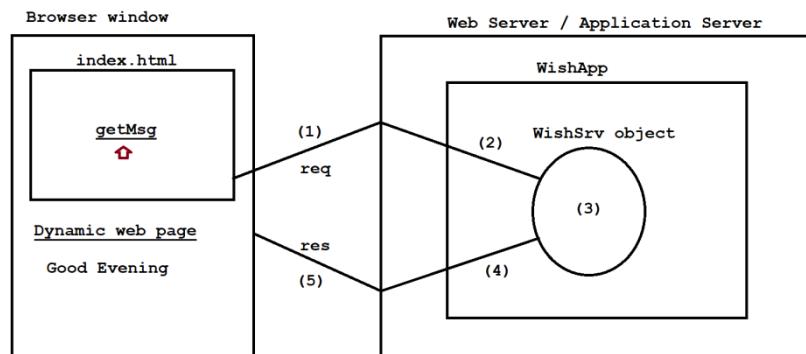


Diagram: servlet3.1

Deployment Directory structure

```
WishApp
|
|---Java Resource
|   |
|   |---src
|   |   |
|   |   |---com.ihub.www
|   |   |   |
|   |   |   |---WishSrv.java
```

```
|  
|---Web Content  
|  
|---index.html  
|  
|---WEB-INF  
|  
|---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

It is never recommended to extends a class with GenericServlet class because it will not give HTTP protocol features.

It is always recommended to extends a class with HttpServlet class because it will give HTTP protocol features.

index.html

```
<center>  
    <h1>  
        <a href="test"> getMsg </a>  
    </h1>  
</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
         xmlns="http://java.sun.com/xml/ns/javaee"  
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
         http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">  
  
<servlet>  
    <servlet-name>WishSrv</servlet-name>  
    <servlet-class>com.ihub.www.WishSrv</servlet-class>  
</servlet>  
<servlet-mapping>  
    <servlet-name>WishSrv</servlet-name>  
    <url-pattern>/test</url-pattern>  
</servlet-mapping>  
  
<welcome-file-list>  
    <welcome-file>index.html</welcome-file>  
</welcome-file-list>  
  
</web-app>
```

TestSrv.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Calendar;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class WishSrv extends HttpServlet
{
    public void service(HttpServletRequest req,HttpServletResponse res) throws
ServletException,IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        Calendar c = Calendar.getInstance();
        //convert time to 24 hours
        int h = c.get(Calendar.HOUR_OF_DAY);

        if(h < 12)
            pw.println("<center><h1>Good Morning</h1></center>");
        else if(h < 16)
            pw.println("<center><h1>Good afternoon</h1></center>");
        else if(h < 20)
            pw.println("<center><h1>Good Evening</h1></center>");
        else
            pw.println("<center><h1>Good Night</h1></center>");

        pw.close();
    }
}
```

Request url

http://localhost:2525/WishApp/test

Example application on HTML based form page to servlet communication

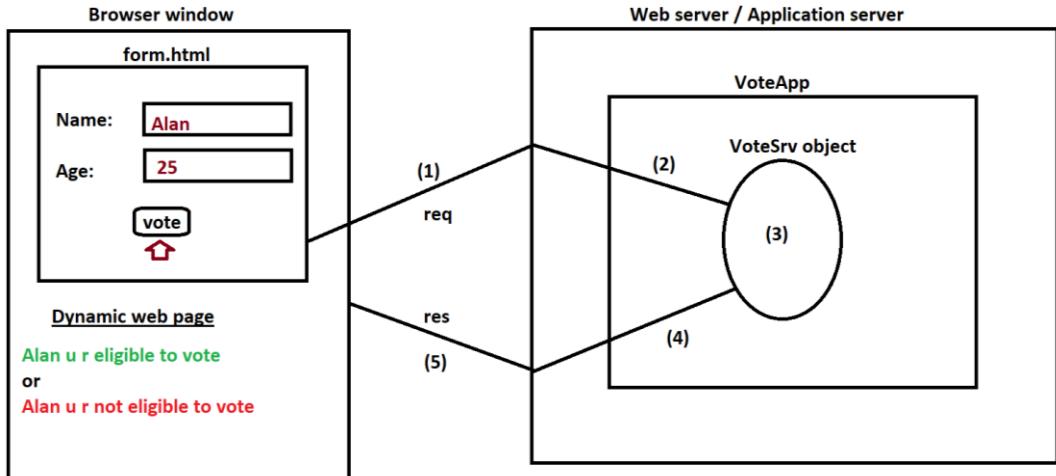


Diagram: servlet4.1

Deployment Directory structure

```
VoteApp
|
|---Java Resources
|   |
|   |---src
|   |   |
|   |   |---com.ihub.www
|   |   |
|   |   |---VoteSrv.java
|---Web Content
|   |
|   |---form.html
|   |
|   |---WEB-INF
|   |   |
|   |   |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.
We can send the request to servlet intwo methodologies.

1) GET Methodology

It will carry limited amount of data.

2) POST methodology

It will carry unlimited amount of data.

While working with HttpServlet class, it is never recommended to work with service(-,-) method because it is not designed according HTTP protocol features. It is always recommended to use doXxx(-,-) methods because they have designed according HTTP protocol features.

We have seven doXxx(-,-) methods as follow.

ex:

- 1) doGet(-,-)
- 2) doPost(-,-)
- 3) doHead(-,-)
- 4) doOption(-,-)
- 5) doTrace(-,-)
- 6) doDelete(-,-)
- 7) doPut(-,-)

prototype of doXxx(-,-)

```
-----  
protected void doGet(HttpServletRequest req, HttpServletResponse res) throws  
ServletException, IOException  
{  
}
```

form.html

```
<form action="test" method="GET">  
    Name: <input type="text" name="t1"/> <br>  
    Age: <input type="text" name="t2"/> <br>  
    <input type="submit" value="vote"/>  
</form>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
         xmlns="http://java.sun.com/xml/ns/javaee"  
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
                           http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">  
  
<servlet>  
    <servlet-name>VoteSrv</servlet-name>  
    <servlet-class>com.ihub.www.VoteSrv</servlet-class>  
</servlet>  
<servlet-mapping>  
    <servlet-name>VoteSrv</servlet-name>  
    <url-pattern>/test</url-pattern>  
</servlet-mapping>  
<welcome-file-list>  
    <welcome-file>form.html</welcome-file>  
</welcome-file-list>  
  
</web-app>
```

VoteSrv.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class VoteSrv extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        //reading form data
        String name = req.getParameter("t1");
        String sage = req.getParameter("t2");

        //converting string age to int age
        int age = Integer.parseInt(sage);

        if (age >= 18)
            pw.println("<center><h1 style='color:green'>" + name + " U r eligible to
vote</h1></center>");
        else
            pw.println("<center><h1 style='color:red'>" + name + " U r not eligible
to vote</h1></center>");

        pw.close();
    }
}

request url
-----
http://localhost:2525/VoteApp/
```

Q) What is the difference between GET and POST methodology?

GET

It is a default methodology.
It will carry limited amount of data.
It sends the request fastly.
It is not suitable for secure data.
It is not suitable for encryption and file uploading.
To process get methodology we will use doGet(-,-) method.

POST

It is not a default methodology.
It will carry unlimited amount of data.
It sends the request bit slow.
It is suitable for secure data.
It is suitable for encryption and file uploading.
To process post methodology we will use doPost(-,-) method.

Servlet to Database Communication

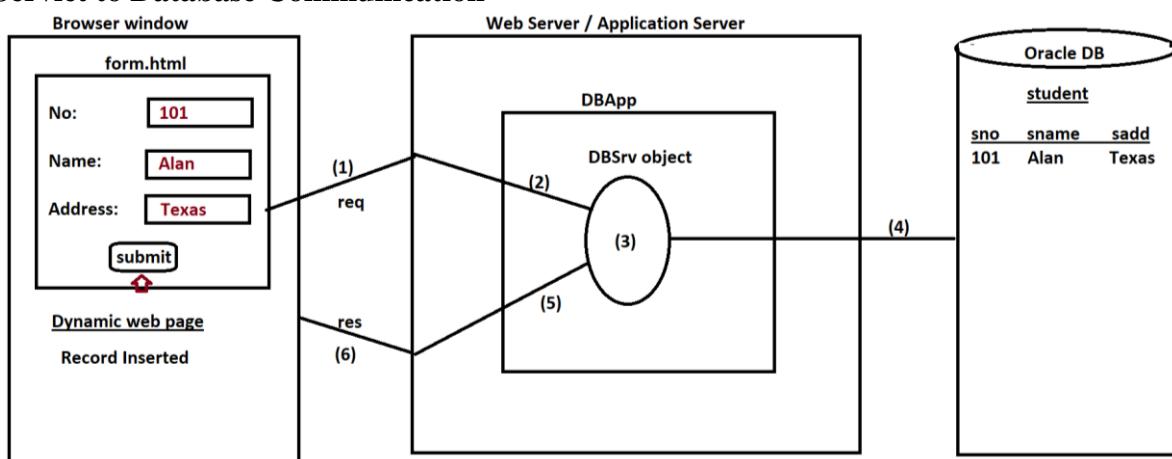


Diagram: servlet4.2

Deployment Directory structure

DBApp

|

|---Java Resources

|
|----src

|

|----com.ihub.www

|

|----DBSrv.java

|

|---Web Content

|
|----form.html

|

|----WEB-INF

|

|----web.xml

|

```
|-----lib  
|  
|---ojdbc14.jar
```

Note:

In above application we need to "servlet-api.jar" and "ojdbc14.jar" file in project build path.

Copy and paste "ojdbc14.jar" file in "WEB-INF/lib" folder seperately.

student table

```
drop table student;  
create table student(sno number(3),sname varchar2(10),sadd varchar2(12));
```

form.html

```
<form action="test" method="GET">  
    No: <input type="text" name="t1"/> <br>  
    Name: <input type="text" name="t2"/> <br>  
    Address: <input type="text" name="t3"/> <br>  
    <input type="submit" value="submit"/>  
</form>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
         xmlns="http://java.sun.com/xml/ns/javaee"  
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
         http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">  
  
<servlet>  
    <servlet-name>DBSrv</servlet-name>  
    <servlet-class>com.ihub.www.DBSrv</servlet-class>  
</servlet>  
<servlet-mapping>  
    <servlet-name>DBSrv</servlet-name>  
    <url-pattern>/test</url-pattern>  
</servlet-mapping>  
  
<welcome-file-list>  
    <welcome-file>form.html</welcome-file>  
</welcome-file-list>  
  
</web-app>
```

DBSrv.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class DBSrv extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        //reading form data
        String sno = req.getParameter("t1");
        int no = Integer.parseInt(sno);
        String name = req.getParameter("t2");
        String add = req.getParameter("t3");

        //insert form data to database
        Connection con = null;
        PreparedStatement ps = null;
        int result = 0;
        String qry = null;
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE", "system",
"admin");
            qry = "insert into student values(?, ?, ?)";
            ps = con.prepareStatement(qry);
            //set the values
            ps.setInt(1, no);
            ps.setString(2, name);
            ps.setString(3, add);
        }
    }
}
```

```

//execute
result=ps.executeUpdate();

if(result==0)
    pw.println("<center>No Record Inserted</center>");
else
    pw.println("<center>Record Inserted</center>");

ps.close();
con.close();

}

catch(Exception e)
{
    pw.println(e);
}
pw.close();
}

}

```

Request url

http://localhost:2525/DBApp/

Form Validation

The process of checking format and pattern of form data is called form validation and such logic is called form validation logic.

We can perform form validation in two ways.

1) Client side form validation

Validation which is performed at client side is called client side form validation.

2) Server side form validation

Validation which is performed at server side is called server side form validation.

Deployment Directory structure

ValidationApp

|

|----Java Resources

|

|-----src

|

|---com.ihub.www

|

|-----FormSrv.java

```

|
|---Web Content
|
|-----form.html
|
|-----validation.js
|
|-----WEB-INF
|
|-----web.xml

```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```

<!DOCTYPE html>
<html>
    <head>
        <!-- add javascript file -->
        <script type="text/javascript" src="validation.js"></script>

    </head>
    <body>
        <form name="myform" action="test" method="GET" onsubmit="return
validate()">

```

Name: <input type="text" name="t1"/>

Age: <input type="text" name="t2"/>

<input type="hidden" name="vflag" value="no"/>

<input type="submit" value="submit"/>

</form>

</body>

</html>

web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
    <servlet>
        <servlet-name>FormSrv</servlet-name>

```

```
<servlet-class>com.ihub.www.FormSrv</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>FormSrv</servlet-name>
    <url-pattern>/test</url-pattern>
</servlet-mapping>
<welcome-file-list>
    <welcome-file>form.html</welcome-file>
</welcome-file-list>
</web-app>
```

validation.js

```
function validate()
{
    var name=document.myform.t1.value;
    var age=document.myform.t2.value;
    document.myform.vflag.value="yes";

    if(name=="")
    {
        alert("Name is mandatory");
        document.myform.t1.focus();
        return false;
    }
    if(age=="")
    {
        alert("Age is mandatory");
        document.myform.t2.focus();
        return false;
    }
    else
    {
        if(isNaN(age))
        {
            alert("Age must be numeric ");
            document.myform.t2.value="";
            document.myform.t2.focus();
            return false;
        }
    }
}

return true;
}
```

FormSrv.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class FormSrv extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        //reading form data
        String name = req.getParameter("t1");
        String sage = req.getParameter("t2");
        String status = req.getParameter("vflag");
        int age = 0;

        if(status.equals("no"))
        {
            if(name == "" || name.length() == 0 || name == null)
            {
                pw.println("<center>Name is mandatory</center>");
                return;
            }
            if(sage == "" || sage.length() == 0 || sage == null)
            {
                pw.println("<center>Age is mandatory</center>");
                return;
            }
            else
            {
                try
                {
                    age = Integer.parseInt(sage);
                }
                catch(NumberFormatException nfe)
```

```

        {
            pw.println("<center>Age must be numeric</center>");
            return;
        }
    }

    if(status.equals("yes"))
    {
        age=Integer.parseInt(sage);
    }

    if(age<18)
        pw.println("<center>U r not eligible to vote</center>");
    else
        pw.println("<center>U r eligible to vote</center>");
    pw.close();
}
}

```

Request url

http://localhost:2525/ValidationApp/

File Uploading

The process of capturing a file from client machine file system and storing in a server machine file system is called file uploading and reverse is called file downloading.

While dealing with matrimonial applications, job portal applications and profile management applications we need to upload and download a file.

There is no specific API in Servlet to perform file uploading.

We need to take the support of third party API called JAVAZOOM API.

JAVAZOOM API comes in zip format and once if we extract then we will get three jar files.

ex:

- uploadbean.jar (main jar file)
- struts.jar (dependent jar file)
- cos.jar (dependent jar file)

We can take file component in a form page as follow.

ex:

<input type="file" name="f1"/>

JAVAZOOM API link

https://drive.google.com/file/d/1LB0WSJvSCCVOgz7xNwyuYtmy_0_TfJzq/view?usp=sharing

Deployment Directory structure

```
UploadApp
|
|---Java Resources
|   |
|   |---src
|   |   |
|   |   |---com.ihub.www
|   |   |   |
|   |   |   |---TestSrv.java
|
|---Web Content
|   |
|   |---form.html
|   |
|   |---WEB-INF
|   |   |
|   |   |---web.xml
|   |   |
|   |   |---lib
|   |   |   |
|   |   |   |---uploadbean.jar
|   |   |   |---struts.jar
|   |   |   |---cos.jar
```

Note:

In above application we need to add "servlet-api.jar" and "uploadbean.jar" file in project build path.

copy and paste javazoom api jar files inside "WEB-INF/lib" folder seperately.

form.html

```
<form action="test" method="POST" enctype="multipart/form-data">
```

```
    File1: <input type="file" name="f1"/> <br>
```

```
    File2: <input type="file" name="f2"/> <br>
```

```
    <input type="submit" value="upload"/>
```

```
</form>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <servlet>
    <servlet-name>TestSrv</servlet-name>
    <servlet-class>com.ihub.www.TestSrv</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>TestSrv</servlet-name>
    <url-pattern>/test</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>form.html</welcome-file>
  </welcome-file-list>

</web-app>

```

TestSrv.java

```

package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import javazoom.upload.MultipartFormDataRequest;
import javazoom.upload.UploadBean;

public class TestSrv extends HttpServlet
{
    protected void doPost(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        //file uploading
        try

```

```
{  
    UploadBean ub=new UploadBean();  
    ub.setFolderstore("C:\\B24");  
    ub.setOverwrite(false);  
  
    MultipartFormDataRequest nreq=new  
    MultipartFormDataRequest(req);  
    ub.store(nreq);  
  
    pw.println("<center>Files are uploaded successfully</center>");  
}  
catch(Exception e)  
{  
    pw.println(e);  
}  
pw.close();  
}  
}
```

Request url

<http://localhost:2525/UploadApp/>

Servlet Filters

Filter is an object which is executed at the time of preprocessing and post processing of the request.

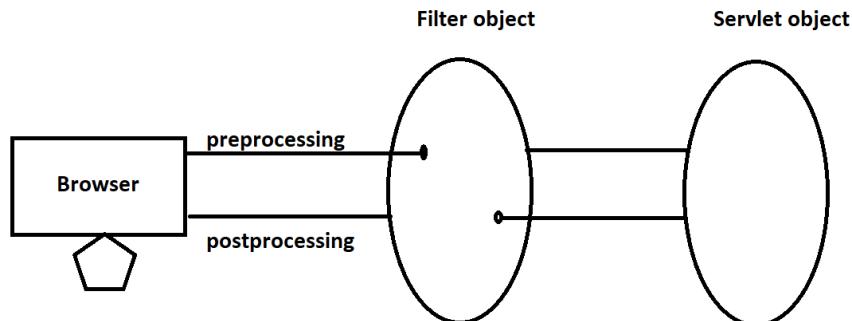


Diagram: servlet6.1

The main purpose of filters are

- 1) We can count number of request coming to the application
 - 2) To perform validations.
 - 3) To perform encryption and Decryption.

Like Servlet, Filter is having its own Filter API.

The `javax.servlet` package contains three interfaces of Filter API.

- 1)Filter
 - 2)FilterChain
 - 3)FilterConfig

1) Filter Interface

For creating any filter, we must and should implement the Filter interface.

Filter interface provides the following 3 life cycle methods for filter.

i) public void init(FilterConfig config)

It is used to initialize the filter.

It invokes only once .

ii) public void doFilter(HttpServletRequest req,HttpServletResponse res,FilterChain chain)

This method is invoked every time when user request to any resources to which the filter is mapped.

IT is used to perform filtering task.

iii) public void destroy()

This method is invoked only once when filter is taken out of the service.

2) FilterChain

It is responsible to invoke the next filter or resource in the chain.

FilterChain contains only one method.

i) public void doFilter(HttpServletRequest req,HttpServletResponse res)

It passes the control to the next filter or resource.

3) FilterConfig

For every filter our servlet container creates FilterConfig object.

It is one per filter.

Deployment Directory structure

FilterApp

|

|---Java Resources

|

|----src

|

|----com.ihub.www

|

|---MyFilter.java

|---MyServlet.java

|

|---Web Content

|

|----index.html

|

|----WEB-INF

|

|----web.xml

Note:

In above application we need to add "servlet-api.jar" file in project build path.

index.html

```
<center>
    <h1>
        <a href="test"> clickMe </a>
    </h1>
</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

    <servlet>
        <servlet-name>MyServlet</servlet-name>
        <servlet-class>com.ihub.www.MyServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>MyServlet</servlet-name>
        <url-pattern>/test</url-pattern>
    </servlet-mapping>

    <filter>
        <filter-name>MyFilter</filter-name>
        <filter-class>com.ihub.www.MyFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>MyFilter</filter-name>
        <url-pattern>/test</url-pattern>
    </filter-mapping>

    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>

</web-app>
```

MyFilter.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class MyFilter implements Filter
{
    @Override
    public void init(FilterConfig config) throws ServletException {
        // TODO Auto-generated method stub

    }

    @Override
    public void doFilter(ServletRequest req, ServletResponse res,
            FilterChain chain) throws IOException, ServletException {

        PrintWriter pw=res.getWriter();
        res.setContentType("text/html");

        pw.println("<center>Filter Invoked Before</center><br>");
        chain.doFilter(req,res);
        pw.println("<center>Filter Invoked After</center><br>");

    }

    @Override
    public void destroy() {
        // TODO Auto-generated method stub

    }
}
```

MyServlet.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class MyServlet extends HttpServlet
{
    protected void doGet(HttpServletRequest req,HttpServletResponse res) throws
ServletException,IOException
    {
        PrintWriter pw=res.getWriter();
        res.setContentType("text/html");

        pw.println("<center>Servlet Executed</center><br>");
    }
}
```

Request url

http://localhost:2525/FilterApp/test

Servlet Life cycle methods

We have three life cycle methods in servlets.

1) public void init(ServletConfig config) throws ServletException

It is used for instantiation event.

This method will execute just before servlet object creation.

2) public void service(ServletRequest req,ServletResponse res) throws ServletException,IOException

It is used for request arrival event.

This method will execute when request goes to servlet program.

3) public void destroy()

It is used for destruction event.

This method will execute just before servlet object destruction.

Deployment Directory structure

```
LifeCycleApp
|
|---Java Resources
|   |
|   |---src
|   |   |
|   |   |---com.ihub.www
|   |   |   |
|   |   |   |----TestSrv.java
|
|---Web Content
|   |
|   |---index.html
|   |
|   |----WEB-INF
|   |   |
|   |   |----web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

index.html

```
<center>
  <h1>
    <a href="test"> click Here </a>
  </h1>
</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  id="WebApp_ID" version="3.0">

  <servlet>
    <servlet-name>TestSrv</servlet-name>
    <servlet-class>com.ihub.www.TestSrv</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>TestSrv</servlet-name>
    <url-pattern>/test</url-pattern>
  </servlet-mapping>
```

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

</web-app>
```

TestSrv.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServlet;

public class TestSrv extends HttpServlet
{
    public void init(ServletConfig config) throws ServletException
    {
        System.out.println("init-method");
    }

    public void service(ServletRequest req, ServletResponse res) throws
ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        pw.println("<center>Service method is called</center>");
        pw.close();
        System.out.println("service-method");
    }

    public void destroy()
    {
        System.out.println("destroy-method");
    }
}
```

Request url

http://localhost:2525/LifeCycleApp/test

ServletConfig object

ServletConfig is an interface which is present in javax.servlet package.

ServletConfig object is created by the web container for every servlet.

ServletConfig object is used to read configuration information from web.xml file.

We can create ServletConfig object as follow.

ex:

```
ServletConfig config=getServletConfig();
```

ServletConfig interface contains following four methods.

1) public String getInitParameter(String name);

It will return parameter value based on specified parameter name.

2) public Enumeration getInitParameterNames();

It will return enumeration of all initialized parameter names.

3) public ServletContext getServletContext();

It will return ServletContext object.

4) public String getServletName();

It will return Servlet name.

Deployment Directory structure

ConfigApp

|

|---Java Resources

|

|

|-----src

|

|---com.ihub.www

|

|-----TestSrv.java

|

|---Web Content

|

|---index.html

|

|-----WEB-INF

|

|-----web.xml

Note:

In above application we need to add "servlet-api.jar" file in project build path.

```
index.html
```

```
-----  
<center>  
    <h1>  
        <a href="test"> click Here </a>  
    </h1>  
</center>
```

```
web.xml
```

```
-----  
<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
         xmlns="http://java.sun.com/xml/ns/javaee"  
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
         http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
```

```
<servlet>  
    <servlet-name>TestSrv</servlet-name>  
    <servlet-class>com.ihub.www.TestSrv</servlet-class>  
    <init-param>  
        <param-name>driver</param-name>  
        <param-value>oracle.jdbc.driver.OracleDriver</param-value>  
    </init-param>  
    <init-param>  
        <param-name>url</param-name>  
        <param-value>jdbc:oracle:thin:@localhost:1521:XE</param-value>  
    </init-param>  
</servlet>
```

```
<servlet-mapping>  
    <servlet-name>TestSrv</servlet-name>  
    <url-pattern>/test</url-pattern>  
</servlet-mapping>
```

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

```
</web-app>
```

TestSrv.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Enumeration;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class TestSrv extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        ServletConfig config = getServletConfig();

        pw.println(config.getInitParameter("driver") + "<br>");
        pw.println(config.getInitParameter("url") + "<br>");

        Enumeration<String> e = config.getInitParameterNames();
        while (e.hasMoreElements())
        {
            String s = e.nextElement();
            pw.println(s + "<br>");
        }

        pw.println(config.getServletName() + "<br>");
        pw.close();
    }
}
```

Request url

http://localhost:2525/ConfigApp/test

ServletContext object

ServletContext is an interface which is present in javax.servlet package.

ServletContext object is created by the web container for every web application.

Servletcontext object is used to read configuration information from web.xml file which is global.

We can create ServletContext object as follow.

ex:

```
ServletContext context=getServletContext();
```

ServletContext contains following methods.

1) public String getInitParameter(String name);

It will return parameter value based on specified parameter name.

2) public Enumeration getInitParameterNames();

It will return enumeration of all initialized parameter names.

Deployment Directory structure

ContextApp

|

|---Java Resources

|

|---src

|

|---com.ihub.www

|

|---TestSrv.java

|

|---Web Content

|

|---index.html

|

|---WEB-INF

|

|---web.xml

Note:

In above application we need to add "servlet-api.jar" file in project build path.

index.html

```
<center>
    <h1>
        <a href="test">Click Here </a>
    </h1>
</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>

    <servlet>
        <servlet-name>TestSrv</servlet-name>
        <servlet-class>com.ihub.www.TestSrv</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>TestSrv</servlet-name>
        <url-pattern>/test</url-pattern>
    </servlet-mapping>

    <context-param>
        <param-name>driver</param-name>
        <param-value>oracle.jdbc.driver.OracleDriver</param-value>
    </context-param>
    <context-param>
        <param-name>url</param-name>
        <param-value>jdbc:oracle:thin:@localhost:1521:XE</param-value>
    </context-param>

</web-app>
```

TestSrv.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Enumeration;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class TestSrv extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        ServletContext context = getServletContext();

        pw.println(context.getInitParameter("driver") + "<br>");
        pw.println(context.getInitParameter("url") + "<br>");

        Enumeration<String> e = context.getInitParameterNames();
        while (e.hasMoreElements())
        {
            String s = e.nextElement();
            pw.println(s + "<br>");
        }

        pw.close();
    }
}
```

Request url

http://localhost:2525/ContextApp

Servlet to Servlet Communication

Servlet to Servlet communication is also known as servlet chaining.

Servlet to servlet communication is possible by using three ways.

- 1)Forwarding the request
- 2)Including the response
- 3)Send Redirection

1) Forwarding the request

In forwarding the request, output of source servlet program will be discarded and only output of destination servlet program goes to browser window as dynamic response.

To forward the request we need to use RequestDispatcher object.

RequestDispatcher is an interface which is present in javax.servlet package.

We can create RequestDispatcher object as follow.

ex:

```
RequestDispatcher rd=req.getRequestDispatcher("url");
rd.forward(req,res);
```

2) Including the response

In cluding the response, output of source servlet program and output of destination servlet program combinely goes to browser window as dynamic response.

For including the response we need to use RequestDispatcher object.

RequestDispatcher is an interface which is present in javax.servlet package.

We can create RequestDispatcher object as follow.

ex:

```
RequestDispatcher rd=req.getRequestDispatcher("url");
rd.include(req,res);
```

Deployment Directory structure

STSApp1

|

|---Java Resources

|

|----src

|

|---com.ihub.www

|

|---TestSrv1.java

|---TestSrv2.java

|

|---Web Content

|

|----form.html

|

|----WEB-INF

|

|---web.xml

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="test1">
    <table align="center">

        <tr>
            <td>UserName:</td>
            <td><input type="text" name="t1"/></td>
        </tr>
        <tr>
            <td>Password:</td>
            <td><input type="password" name="t2"/></td>
        </tr>
        <tr>
            <td><input type="reset" value="reset"/></td>
            <td><input type="submit" value="submit"/></td>
        </tr>
    </table>
</form>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

    <servlet>
        <servlet-name>TestSrv1</servlet-name>
        <servlet-class>com.ihub.www.TestSrv1</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>TestSrv1</servlet-name>
        <url-pattern>/test1</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>TestSrv2</servlet-name>
        <servlet-class>com.ihub.www.TestSrv2</servlet-class>
    </servlet>
    <servlet-mapping>
```

```

<servlet-name>TestSrv2</servlet-name>
<url-pattern>/test2</url-pattern>
</servlet-mapping>

<welcome-file-list>
    <welcome-file>form.html</welcome-file>
</welcome-file-list>

</web-app>

```

TestSrv1.java

```

package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class TestSrv1 extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        //reading form data
        String name = req.getParameter("t1");
        String pwd = req.getParameter("t2");

        if(pwd.equals("admin"))
        {
            RequestDispatcher rd = req.getRequestDispatcher("test2");
            rd.forward(req, res);
        }
        else
        {
            pw.println("<center><b style='color:red'>Sorry! Incorrect username or
password</b></center>");
            RequestDispatcher rd = req.getRequestDispatcher("form.html");
        }
    }
}

```

```

        rd.include(req,res);
    }

    pw.close();
}
}

```

TestSrv2.java

```

package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class TestSrv2 extends HttpServlet
{
    protected void doGet(HttpServletRequest req,HttpServletResponse res) throws
ServletException,IOException
    {
        PrintWriter pw=res.getWriter();
        res.setContentType("text/html");

        pw.println("<center>Login Successfully!!!</center>");

        pw.close();
    }
}

```

Request url

http://localhost:2525/STSApp1/

3) Send Redirection

It is used to forward the request to web application which is present in same server or different server.

Send Redirection by using sendRedirect() method HttpServletResponse object.

ex:

```
res.sendRedirect("url");
```

Send redirection will work inside the server and outside the server.

It always sends a new request.

It uses browser window to send the request.

Deployment Directory structure

```
STSApp2
|
|---Java Resources
|   |
|   |---src
|   |   |
|   |   |---com.ihub.www
|   |   |   |
|   |   |   |---TestSrv.java
|
|---Web Content
|   |
|   |---index.html
|   |
|   |---WEB-INF
|   |   |
|   |   |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

index.html

```
<center>
<h1>
    <a href="test?t1=flights"> Flights </a> <br><br>
    <a href="test?t1=hotels"> Hotels </a> <br><br>
    <a href="test?t1=railways"> Trains </a> <br><br>
</h1>
</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
    <servlet>
        <servlet-name>TestSrv</servlet-name>
        <servlet-class>com.ihub.www.TestSrv</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>TestSrv</servlet-name>
        <url-pattern>/test</url-pattern>
    </servlet-mapping>
```

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

</web-app>
```

TestSrv.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class TestSrv extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        String val = req.getParameter("t1");

        res.sendRedirect("https://www.makemytrip.com/" + val);

        pw.close();
    }
}
```

Request url

http://localhost:2525/STSApp2/

Q) How to enable <load-on-startup> and what happens if we enable <load-on-startup>?

We can enable <load-on-startup> inside web.xml file.

If we enable <load-on-startup> then our servlet container will create servlet object during the server startup or during the deployment of web application.

web.xml

```
<web-app>
    <servlet>
        <servlet-name>TestSrv2</servlet-name>
        <servlet-class>com.ihub.www.TestSrv2</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>TestSrv2</servlet-name>
        <url-pattern>/test2</url-pattern>
    </servlet-mapping>
</web-app>
```

Stateless Behaviour of web application

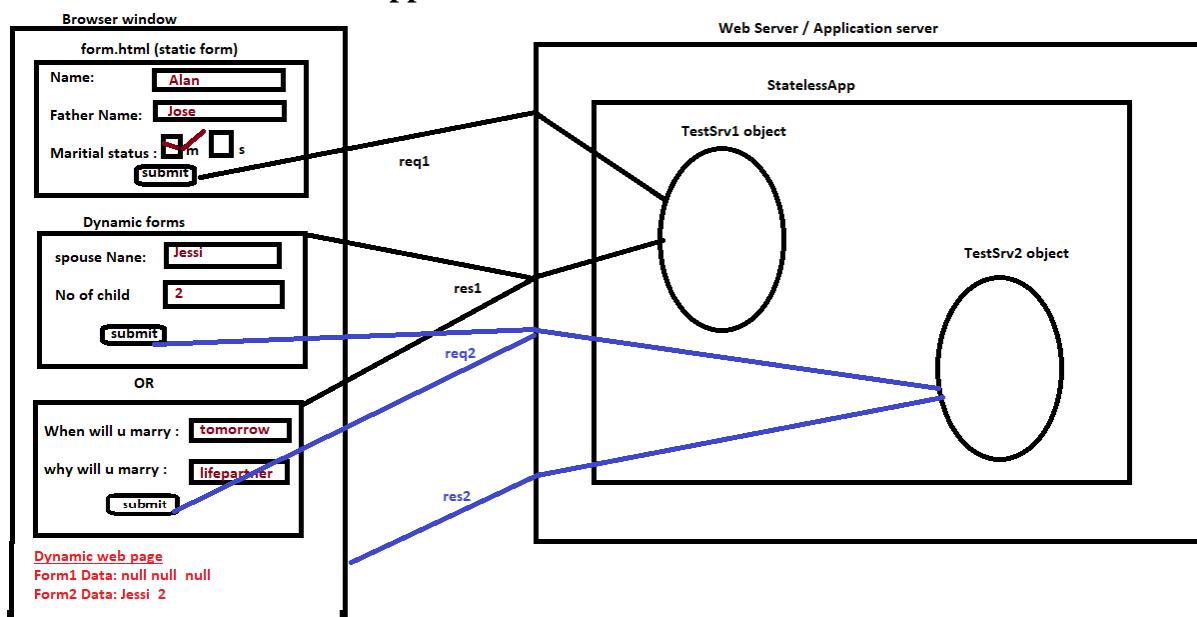


Diagram: servlet9.1

In above diagram demonstrate stateless behaviour of web application.

Stateless web application means while working with current request we can't access previous request data.

Our HTTP protocol is stateless which makes our web application also stateless.

To overcome this limitation we need to use Session Tracking.

Deployment Directory structure

```
StatelessApp
|
|---Java Resources
|   |---src
|   |   |---com.ihub.www
|   |   |   |---TestSrv1.java
|   |   |   |---TestSrv2.java
|---Web Content
|   |---form.html
|   |---WEB-INF
|   |   |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="test1">
    Name: <input type="text" name="t1"/> <br>
    Father Name : <input type="text" name="t2"/> <br>
    Marital Status :
    <input type="checkbox" name="t3" value="married"/> MARRIED
    <input type="checkbox" name="t3" value="single"/> SINGLE
    <br>
    <input type="submit" value="submit"/>
</form>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

    <servlet>
        <servlet-name>TestSrv1</servlet-name>
        <servlet-class>com.ihub.www.TestSrv1</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
```

```

<servlet-mapping>
    <servlet-name>TestSrv1</servlet-name>
    <url-pattern>/test1</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>TestSrv2</servlet-name>
    <servlet-class>com.ihub.www.TestSrv2</servlet-class>
    <load-on-startup>2</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>TestSrv2</servlet-name>
    <url-pattern>/test2</url-pattern>
</servlet-mapping>

<welcome-file-list>
    <welcome-file>form.html</welcome-file>
</welcome-file-list>

</web-app>

```

TestSrv1.java

```

package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class TestSrv1 extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        // reading form1 data

        String name = req.getParameter("t1");
    }
}

```

```

String fname=req.getParameter("t2");
String ms=req.getParameter("t3");

if(ms.equals("married"))
{
    pw.println("<form action='test2'>");
    pw.println("Spouse Name :<input type='text' name='f2t1'> <br>");
    pw.println("No of Children: <input type='text' name='f2t2'> <br>");
    pw.println("<input type='submit' value='submit'>");
    pw.println("</form>");
}
else
{
    pw.println("<form action='test2'>");
    pw.println("When will u marry :<input type='text' name='f2t1'>
<br>");
    pw.println("Why will u marry: <input type='text' name='f2t2'> <br>");
    pw.println("<input type='submit' value='submit'>");
    pw.println("</form>");
}
pw.close();
}
}

```

TestSrv2.java

```

package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class TestSrv2 extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        PrintWriter pw=res.getWriter();
        res.setContentType("text/html");

        //reading form1 data
    }
}

```

```

String name=req.getParameter("t1");
String fname=req.getParameter("t2");
String ms=req.getParameter("t3");

//reading form2 data
String val1=req.getParameter("f2t1");
String val2=req.getParameter("f2t2");

pw.println("Form 1 Data :" +name+ " "+fname+" "+ms+"<br>");
pw.println("Form 2 Data :" +val1+ " "+val2+"<br>");

pw.close();
}
}

```

Request url

http://localhost:2525/StatelessApp/

Session

The process of continue and related operations perform on web application with multiple request and response is called session.

ex:

login to facebook and logout from facebook is one session.
starting of java class and ending of java class is one session.

Session Tracking / Session Management

Session tracking makes our web application as statefull web application even thoughour HTTP protocol is stateless.

In stateless web application, no web resource program can access previous request data while processing the current request during a session.

In statefull web application , all web resource programs can access previous request data while processing the current request during a session.

Session tracking can be perform in four techniques.

- 1) Using hidden box fields
- 2) HttpCookies
- 3) HttpSession with Cookies
- 4) URL Rewriting

3) HttpSession with Cookies.

HttpSession is an interface which is present in javax.servlet package.

HttpSession always create a session ID for every request to identify where enduser is a existing user or old user.

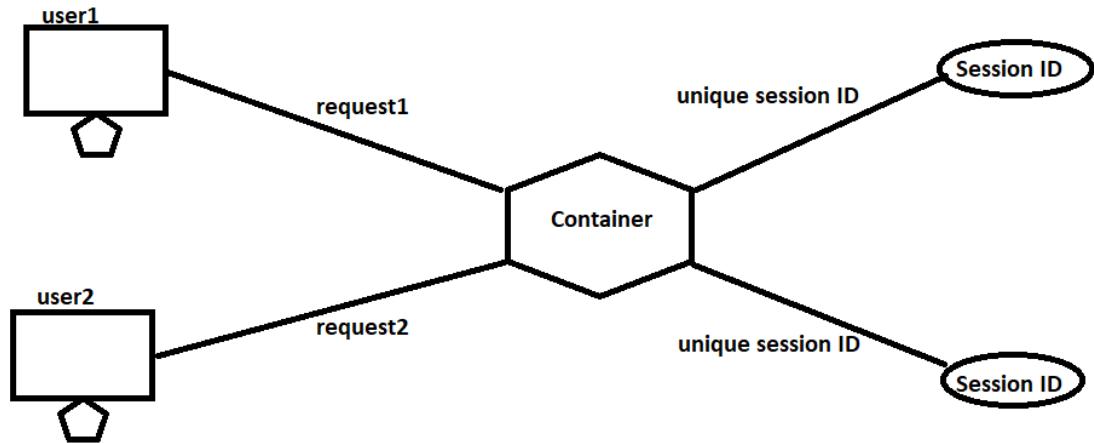


Diagram: servlet9.2

HttpSession object is to perform following things.

- 1) Bind the object
- 2) It will manipulate the data present HttpSession.

Deployment Directory structure

```

SessionTrackingApp
|
|----Java Resources
|   |
|   |----src
|   |   |
|   |   |----com.ihub.www
|   |   |   |
|   |   |   |----TestSrv1.java
|   |   |   |----TestSrv2.java
|
|----Web Content
|   |
|   |----form.html
|   |
|   |----WEB-INF
|   |   |
|   |   |----web.xml

```

Note:

In above application , we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="test1" method="GET">
```

Name: <input type="text" name="t1"/>

Father Name : <input type="text" name="t2"/>

Marital Status :

<input type="checkbox" name="t3" value="married"/>MARRIED
<input type="checkbox" name="t3" value="single"/>SINGLE

<input type="submit" value="submit"/>

</form>

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <servlet>
    <servlet-name>TestSrv1</servlet-name>
    <servlet-class>com.ihub.www.TestSrv1</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>TestSrv1</servlet-name>
    <url-pattern>/test1</url-pattern>
  </servlet-mapping>

  <servlet>
    <servlet-name>TestSrv2</servlet-name>
    <servlet-class>com.ihub.www.TestSrv2</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>TestSrv2</servlet-name>
    <url-pattern>/test2</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>form.html</welcome-file>
  </welcome-file-list>

</web-app>
```

TestSrv1.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class TestSrv1 extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        String name = req.getParameter("t1");
        String fname = req.getParameter("t2");
        String ms = req.getParameter("t3");

        HttpSession session = req.getSession(true);
        session.setAttribute("pname", name);
        session.setAttribute("pfname", fname);
        session.setAttribute("pms", ms);

        if (ms.equals("married"))
        {
            pw.println("<form action='test2'>");
            pw.println("spouse Name : <input type='text' name='f2t1'> <br>");
            pw.println("No of Child : <input type='text' name='f2t2'> <br>");
            pw.println("<input type='submit' value='submit'>");
            pw.println("</form>");
        }
        else
        {
            pw.println("<form action='test2'>");
            pw.println("When will u marry : <input type='text' name='f2t1'>
<br>");
            pw.println("Why will u marry : <input type='text' name='f2t2'>
<br>");
        }
    }
}
```

```

        pw.println("<input type='submit' value='submit' />");
        pw.println("</form>");
    }
    pw.close();
}
}

```

TestSrv2.java

```

package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class TestSrv2 extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
    ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        //reading form 1 data
        HttpSession session = req.getSession(false);
        String name = (String) session.getAttribute("pname");
        String fname = (String) session.getAttribute("pfname");
        String ms = (String) session.getAttribute("pms");
        //reading form 2 data
        String val1 = req.getParameter("f2t1");
        String val2 = req.getParameter("f2t2");

        pw.println("Form 1 data :" + name + " " + fname + " " + ms + "<br>");
        pw.println("Form 2 data :" + val1 + " " + val2 + "<br>");

        pw.close();
    }
}

```

Request url

http://localhost:2525/SessionTrackingApp/

Life cycle methods of JSP

JSP contains three life cycle methods.

1) `_jspInit()`

It is used for instantiation event.

This method will execute just before JES class object creation.

Here JES stands for Java Equivalent Servlet.

2) `_jspService()`

It is used for request arrival event.

This method will execute when request goes to JSP program.

3) `_jspDestroy()`

It is used for destruction event.

This method will execute just before JES class object destruction.

Phases in JSP

We have two phases in JSP.

1) Translation phase

In translation phase our JSP program converts to JES class.

2) Request Processing phase

In request processing phase our JES class will be executed and result will send to browser window as dynamic response.

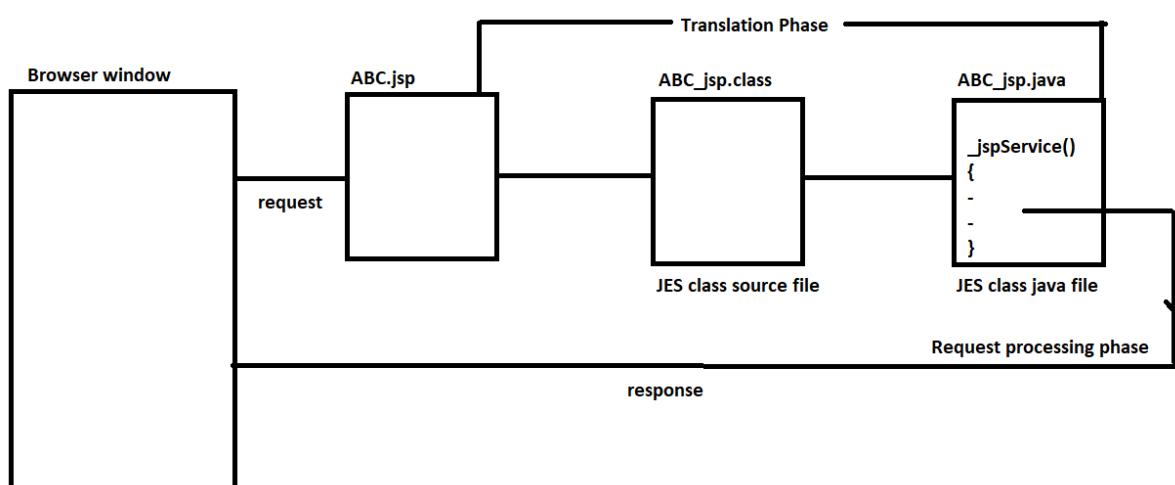


Diagram: jsp2.1

Q) How to enable `<load-on-startup>` and what happens if we enable `<load-on-startup>`?

We can enable `<load-on-startup>` inside web.xml file.

web.xml

```
<web-app>
  <servlet>
    <servlet-name>ABC</servlet-name>
    <jsp-file>/ABC.jsp</jsp-file>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
```

```
<servlet-name>ABC</servlet-name>
<url-pattern>/test</url-pattern>
</servlet-mapping>
</web-app>
```

If we enable `<load-on-startup>` then translation phase will be perform during server startup or during the deployment of web application.

It means our JES class object will be ready before we give the first request.

JSP Tags/Elements

We have following tags in JSP.

1) Scripting Tags

i) scriptlet tag

ex: `<% code %>`

ii) expression tag

ex: `<%= code %>`

iii) declaration tag

ex: `<%! code %>`

2) Directive Tags

i) page directive tag

ex: `<%@page attribute=value %>`

ii) include directive tag

ex: `<%@include attribute=value %>`

3) Standard Tags

ex:

```
<jsp:include>
<jsp:forward>
<jsp:useBean>
<jsp:setProperty>
<jsp:getProperty>
and etc.
```

4) Comments

ex: `<%-- comment here --%>`

i) scriptlet tag

A scriptlet tag is used to declare java code.

syntax: `<% code %>`

Deployment Directory structure

JspApp2

|

|---Java Resources

|

|---Web Content

|

```
|---form.html  
|---process.jsp  
|  
|---WEB-INF  
|  
|---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="process.jsp">  
    Name: <input type="text" name="t1"/> <br>  
    <input type="submit" value="submit"/>  
</form>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
         xmlns="http://java.sun.com/xml/ns/javaee"  
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
         http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">  
  
<welcome-file-list>  
    <welcome-file>form.html</welcome-file>  
</welcome-file-list>  
  
</web-app>
```

process.jsp

```
<center>  
    <h1>  
        <%  
            String name=request.getParameter("t1");  
            out.println("Welcome :" + name);  
        %>  
    </h1>  
</center>
```

request url

http://localhost:2525/JspApp2/

ii) expression tag

The code which is written in expression will return to the output stream of a response so we don't need to write out.println() to print the data.

syntax: <%= code %>

Note:

Expression tag does not allow semicolon.

Deployment Directory structure

```
JspApp2
|
|---Java Resources
|
|---Web Content
    |
    |---form.html
    |---process.jsp
    |
    |---WEB-INF
        |
        |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="process.jsp">
    Name: <input type="text" name="t1"/> <br>
    <input type="submit" value="submit"/>
</form>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

    <welcome-file-list>
        <welcome-file>form.html</welcome-file>
    </welcome-file-list>

</web-app>
```

```
process.jsp
<center>
    <h1>
        <%
            String name=request.getParameter("t1");
        %>
        <%= "Hello :" +name %>
    </h1>
</center>
```

request url

http://localhost:2525/JspApp2/

iii) declaration tag

Declaration tag is used to declare fields and methods.

syntax: <%! code %>

Deployment Directory Structure

```
JspApp3
|
|---Java Resources
|
|---Web Content
|   |
|   |---index1.jsp
|   |---index2.jsp
|   |
|   |---WEB-INF
|       |
|       |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

```
index1.jsp
<center>
    <h1>
        <%!
            int data=100;
        %>
        <%= "The value is =" +data %>
    </h1>
</center>
```

index2.jsp

```
<center>
<h1>
<%!
    int cube(int n)
    {
        return n*n*n;
    }
%>
<%= "Cube of a given number is =" +cube(5) %>
</h1>
</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

</web-app>
```

request url

http://localhost:2525/JspApp3/index1.jsp
 http://localhost:2525/JspApp3/index2.jsp

Exception Handling in JSP

Runtime errors are called exceptions.

Exception may rise any time in our application so handling the exceptions is a safer side for the programmer.

In JSP, There are two ways to handle exceptions.

- 1) By using `errorPage` and `isErrorPage` attributes of page directive tag
- 2) By using `<error-page>` element in web.xml file

1) By using `errorPage` and `isErrorPage` attributes of page directive tag

Deployment Directory structure

```
JspApp4
|
|--Java Resources
|
|--Web Content
    |
    |---form.html
    |---process.jsp
```

```
|---error.jsp  
|  
|---WEB-INF  
|  
|---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="process.jsp">  
    No1: <input type="text" name="t1"/> <br>  
    No2: <input type="text" name="t2"/> <br>  
    <input type="submit" value="divide"/>  
</form>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
         xmlns="http://java.sun.com/xml/ns/javaee"  
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
         http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">  
  
    <welcome-file-list>  
        <welcome-file>form.html</welcome-file>  
    </welcome-file-list>  
  
</web-app>
```

process.jsp

```
<%@page errorPage="error.jsp" %>  
<%  
    String sno1=request.getParameter("t1");  
    String sno2=request.getParameter("t2");  
    int a=Integer.parseInt(sno1);  
    int b=Integer.parseInt(sno2);  
    int c=a/b;  
%>  
<%= "Division of two numbers is =" +c %>
```

error.jsp

```
<%@page isErrorPage="true" %>  
<b><i>  
    Sorry Exception Occured!! <br>
```

```
<%= exception %>
</i></b>
```

Request url

http://localhost:2525/JspApp4/

2) By using <error-page> element in web.xml file

This approach is recommended to use because we don't need to defining errorPage element in each jsp file. Defining single entry in web.xml file will handle all types of exceptions.

Deployment Directory structure

```
JspApp4
|
|--Java Resources
|
|--Web Content
    |
    |---form.html
    |---process.jsp
    |---error.jsp
    |
    |---WEB-INF
        |
        |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="process.jsp">
    No1: <input type="text" name="t1"/> <br>
    No2: <input type="text" name="t2"/> <br>
    <input type="submit" value="divide"/>
</form>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

    <error-page>
        <exception-type>java.lang.Exception</exception-type>
        <location>/error.jsp</location>
    </error-page>
```

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

</web-app>
```

process.jsp

```
<%
    String sno1=request.getParameter("t1");
    String sno2=request.getParameter("t2");
    int a=Integer.parseInt(sno1);
    int b=Integer.parseInt(sno2);
    int c=a/b;
%>
<%= "Division of two numbers is =" +c %>
```

error.jsp

```
<%@page isErrorPage="true" %>
<b><i>
    Sorry Exception Occured!! <br>
<%= exception %>
</i></b>
```

Request url

http://localhost:2525/JspApp4/

JSP to Database Communication

Deployment Directory structure

```
JspApp5
|
|---Java Resources
|
|---Web Content
    |
    |---form.html
    |---process.jsp
    |
    |---WEB-INF
        |
        |---web.xml
        |
```

```
|-----lib  
|  
|---ojdbc14.jar
```

Note:

In above application we need to add "servlet-api.jar" and "ojdbc14.jar" file in project build path.

Copy and paste "ojdbc14.jar" file in project build path.

form.html

```
<form action="process.jsp">  
    No: <input type="text" name="t1"/> <br>  
    Name: <input type="text" name="t2"/> <br>  
    Address: <input type="text" name="t3"/> <br>  
    <input type="submit" value="submit"/>  
</form>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<web-app  
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
        xmlns="http://java.sun.com/xml/ns/javaee"  
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
        http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">  
  
<welcome-file-list>  
    <welcome-file>form.html</welcome-file>  
</welcome-file-list>  
  
</web-app>
```

process.jsp

```
<%@page import="java.sql.*" buffer="8kb" language="java" %>  
<%  
    String sno=request.getParameter("t1");  
    int no=Integer.parseInt(sno);  
  
    String name=request.getParameter("t2");  
  
    String add=request.getParameter("t3");  
  
    //insert the data into database  
    Connection con=null;  
    PreparedStatement ps=null;  
    String qry=null;  
    int result=0;
```

```

try
{
    Class.forName("oracle.jdbc.driver.OracleDriver");

    con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system",
"admin");
    qry="insert into student values(?, ?, ?)";
    ps=con.prepareStatement(qry);
    //set the values
    ps.setInt(1,no);
    ps.setString(2,name);
    ps.setString(3,add);

    //execute
    result=ps.executeUpdate();

    if(result==0)
        out.println("No Record inserted");
    else
        out.println("Record inserted");

    ps.close();
    con.close();
}
catch(Exception e)
{
    out.println(e);
}

%>

```

Request url

http://localhost:2525/JspApp5/

Actions Tags

Action tags are used to perform particular task.

Action tags are used to control the web pages and uses java beans.

Action tags are executed dynamically at runtime.

Action tags contain standard tags but we don't have xml tags.

Action tags are divided into two types.

- 1) Standard Action tags
- 2) Custom Action tags

1) Standard Action tags

Built-In action tags are called standard action tags.

We have following list of standard action tags.

ex:

```
<jsp:include>
<jsp:forward>
<jsp:useBean>
<jsp:setProperty>
<jsp:getProperty>
and etc.
```

Action forward tag

In action forward the output of source jsp program will be discarded and output of destination jsp program goes to browser window as dynamic response.

It internally uses servlet API functionality called rd.forward(req,res).

syntax: <jsp:forward page="page_name"/>

Deployment Directory structure

```
JspApp6
|
|---Java Resources
|
|---Web Content
    |
    |--A.jsp
    |--B.jsp
    |
    |---WEB-INF
        |
        |--web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

A.jsp

```
<b><i> Begining of A.jsp</i></b>
<br>
<jsp:forward page="B.jsp"/>
<br>
<b><i>Ending of A.jsp</i></b>
```

B.jsp

```
<b><i>This is B.jsp</i></b>
```

```

web.xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <welcome-file-list>
    <welcome-file>A.jsp</welcome-file>
  </welcome-file-list>

</web-app>

```

Request url

http://localhost:2525/JspApp6/

Action include

In action include the output of source jsp program and destination jsp program combinely goes to browser window as dynamic response.

It internally uses servlet API functionality called rd.include(req,res);

syntax:

```
<jsp:include page="page_name"/>
```

Deployment Directory structure

```

JspApp6
|
|---Java Resources
|
|---Web Content
  |
  |--A.jsp
  |--B.jsp
  |
  |---WEB-INF
    |
    |--web.xml

```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

A.jsp

```
<b><i> Begining of A.jsp</i></b>
<br>
```

```
<jsp:include page="B.jsp"/>
<br>
<b><i>Ending of A.jsp</i></b>
```

B.jsp

```
<b><i>This is B.jsp</i></b>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <welcome-file-list>
    <welcome-file>A.jsp</welcome-file>
  </welcome-file-list>

</web-app>
```

Request url

```
http://localhost:2525/JspApp6/
```

JSP to Java Bean Communication

JSP to java bean communication is possible by using three tags.

1) <jsp:useBean> tag

It is used to locate and create bean class object.

2) <jsp:setProperty> tag

It is used to set the value to bean object and calls setter methods.

3) <jsp:getProperty> tag

It is used to get the value from bean object and calls getter methods.

Note:

All above tags are independent tags.

ex:1

Deployment Directory structure

```
JspApp7
```

```
|
```

```
--Java Resources
```

```
|
```

```
-----src
```

```
|
```

```
---com.ihub.www
```

```

|---CubeNumber.java
|---Web Content
|   |
|   |----process.jsp
|   |
|   |----WEB-INF
|       |
|       |----web.xml

```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

CubeNumber.java

```

package com.ihub.www;

public class CubeNumber
{
    public int cube(int n)
    {
        return n*n*n;
    }
}

```

process.jsp

```

<jsp:useBean id="cn" class="com.ihub.www.CubeNumber"></jsp:useBean>
<%= "Cube of a given number is =" + cn.cube(5) %>

```

web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns="http://java.sun.com/xml/ns/javaee"
          xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
          http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

    <welcome-file-list>
        <welcome-file>process.jsp</welcome-file>
    </welcome-file-list>

</web-app>

```

Request url

http://localhost:2525/JspApp7/

ex:2

Deployment Directory structure

```
JspApp8
|
|---Java Resources
|   |
|   |---src
|   |   |
|   |   |---com.ihub.www
|   |   |   |
|   |   |   |---User.java
|---Web Content
|   |
|   |---form.html
|   |---process.jsp
|   |
|   |---WEB-INF
|   |   |
|   |   |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" in project build path.

form.html

```
<form action="process.jsp">

    UserName: <input type="text" name="username"/> <br>
    Password: <input type="password" name="password"/> <br>
    Email: <input type="text" name="email"/> <br>
    <input type="submit" value="submit"/>
```

</form>

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns="http://java.sun.com/xml/ns/javaee"
           xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

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

```
</web-app>
```

User.java

```
package com.ihub.www;

public class User
{
    private String username;
    private String password;
    private String email;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

process.jsp

```
<jsp:useBean id="u" class="com.ihub.www.User"></jsp:useBean>
<jsp:setProperty property="*" name="u"/>
Records are : <br>
<jsp:getProperty property="username" name="u"/> <br>
<jsp:getProperty property="password" name="u"/> <br>
<jsp:getProperty property="email" name="u"/> <br>
```

Request url

```
http://localhost:2525/JspApp8/
```

2) Custom tags in JSP

To create a custom tag in JSP we need to use taglib directory.

We can declare taglib directory as follow.

ex: <%@taglib uri="uriofthetaglibrary" prefix="prefixoftaglibrary" %>

Deployment Directory structure

```
JspApp9
|
|---Java Resources
|   |
|   |---src
|   |   |
|   |   |---com.ihub.www
|   |   |   |
|   |   |   |---CubeNumber.java
|
|---Web Content
|   |
|   |---process.jsp
|   |
|   |---WEB-INF
|   |   |
|   |   |---web.xml
|   |   |---mytags.tld
|   |   |
|   |   |---lib
|   |   |   |
|   |   |   |---jsp-api.jar
```

Note:

In above application we need to add "servlet-api.jar" and "jsp-api.jar" file in project build path.

Copy and paste "jsp-api.jar" file inside "WEB-INF/lib" folder sperately.

process.jsp

```
<%@taglib uri="/WEB-INF/mytags.tld" prefix="ihub" %>
Cube of a given number is : <ihub:cube number="5"/>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
  <welcome-file-list>
```

```

<welcome-file>process.jsp</welcome-file>
</welcome-file-list>

</web-app>

mytags.tld
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
    PUBLIC "-//Sun Microsystems, Inc./DTD JSP Tag Library 1.2//EN"
    "http://java.sun.com/j2ee/dtd/web-jsptaglibrary_1_2.dtd">

<taglib>

<tlib-version>1.0</tlib-version>
<jsp-version>1.2</jsp-version>
<short-name>simple</short-name>
<uri>http://tomcat.apache.org/example-taglib</uri>

<tag>
    <name>cube</name>
    <tag-class>com.ihub.www.CubeNumber</tag-class>
    <attribute>
        <name>number</name>
        <required>true</required>
    </attribute>
</tag>

</taglib>

```

CubeNumber.java

```

package com.ihub.www;

import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.TagSupport;

public class CubeNumber extends TagSupport
{
    private int number;

    //setter method
    public void setNumber(int number)

```

```

{
    this.number=number;
}

public int doStartTag()throws JspException
{
    try
    {
        JspWriter out=pageContext.getOut();
        out.println(number*number*number);
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }

    return SKIP_BODY;
}
}

```

Request url

http://localhost:2525/JspApp9/

MVC in JSP

MVC stands for Model View Controller.

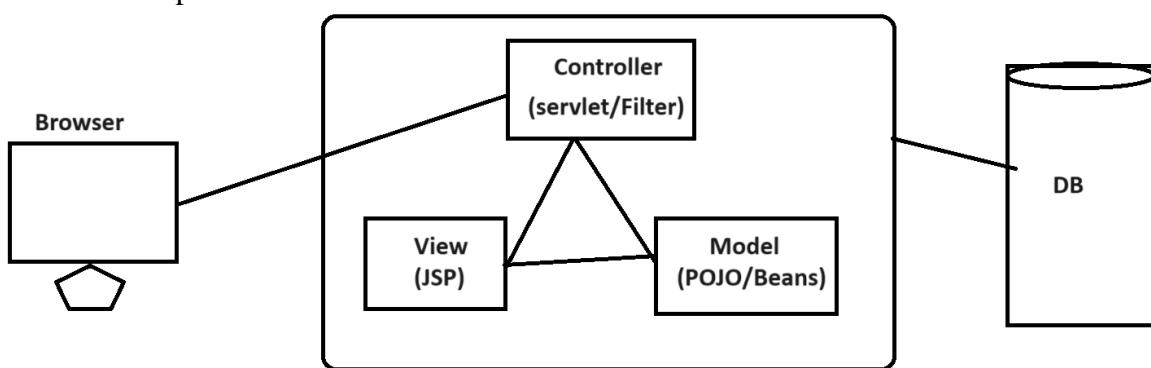
It is one of the design pattern which separates business logic , persistence logic and data.

Controller is an interface between Model and View.

Controller will intercept all incoming request.

Model contains data and sometimes it contains business logic.

View contains presentation i.e UI.



MVC Architecture

Diagram: jsp5.1

Deployment Directory structure

```
MVCApp
|
|---Java Resources
|   |
|   |---src
|   |   |
|   |   |---com.ihub.www
|   |   |   |
|   |   |   |---LoginBean.java
|   |   |   |---LoginSrv.java
|
|---Web Content
|   |
|   |---form.html
|   |---view.jsp
|   |---error.jsp
|   |
|   |---WEB-INF
|   |   |
|   |   |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="test">
    UserName: <input type="text" name="username"/> <br>
    Password: <input type="password" name="password"/> <br>
    <input type="submit" value="login"/>
</form>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

    <servlet>
        <servlet-name>LoginSrv</servlet-name>
        <servlet-class>com.ihub.www.LoginSrv</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginSrv</servlet-name>
```

```
<url-pattern>/test</url-pattern>
</servlet-mapping>

<welcome-file-list>
    <welcome-file>form.html</welcome-file>
</welcome-file-list>

</web-app>
```

LoginBean.java

```
package com.ihub.www;

public class LoginBean
{
    private String username;
    private String password;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

LoginSrv.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```

public class LoginSrv extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        //reading form data
        String uname = req.getParameter("username");
        String pass = req.getParameter("password");

        //create login bean object and set the values
        LoginBean lb = new LoginBean();
        lb.setUsername(uname);
        lb.setPassword(pass);

        req.setAttribute("bean", lb);

        if (pass.equals("admin"))
        {
            RequestDispatcher rd = req.getRequestDispatcher("view.jsp");
            rd.forward(req, res);
        }
        else
        {
            RequestDispatcher rd = req.getRequestDispatcher("error.jsp");
            rd.forward(req, res);
        }

        pw.close();
    }
}

```

view.jsp

```

<%@page import="com.ihub.www.LoginBean" %>
<%
    LoginBean lb = (LoginBean) request.getAttribute("bean");
%>
Details are : <br>

<%= lb.getUsername() %> <br>
<%= lb.getPassword() %> <br>

```

```

error.jsp
<b><i>
    <font color="red">Sorry! incorrect username or password</font>
</i></b>
<%@include file="form.html" %>

```

request url

http://localhost:2525/MVCApp/

Implicit objects in JSP

Object which can be used directly without any configuration is called implicit object. Implicit object is created by the web container which is available for every JSP page. JSP contains 9 implicit objects as shown below.

ex:

<u>Object</u>	<u>Type</u>
out	JspWriter
request	HttpServletRequest
response	HttpServletResponse
config	ServletConfig
application	ServletContext
session	HttpSession
pageContext	pageContext
page	Object
exception	Throwable

response object

In jsp, response is a implicit object of type HttpServletResponse.

It can be used to add or manipulate response such as redirect response or another resources,send error and etc.

Deployment Directory

```

JspApp10
|
|---Java Resources
|
|---Web Content
    |
    |---index.html
    |
    |---process.jsp
    |
    |-----WEB-INF
        |
        |---web.xml

```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<center>
    <h1>
        <a href="process.jsp"> Facebook Page </a>
    </h1>
</center>
```

process.jsp

```
<%
    response.sendRedirect("http://www.facebook.com/login");
%>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

    <welcome-file-list>
        <welcome-file>form.html</welcome-file>
    </welcome-file-list>
</web-app>
```

request url

http://localhost:2525/JspApp10/

config object

It is an implicit object of type ServletConfig.

The config object is created by web container for each jsp page.

This object is used to read initialized parameters for a particular jsp page.

Deployment Directory

```
JspApp11
|
|---Java Resources
|
|---Web Content
```

```
|  
|---index.html  
|  
|---process.jsp  
|  
|-----WEB-INF  
|  
|---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

index.html

```
<center>  
    <h1>  
        <a href="test"> clickMe </a>  
    </h1>  
</center>
```

process.jsp

```
<%  
    String value=config.getInitParameter("driver");  
%>  
<%= value %>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
         xmlns="http://java.sun.com/xml/ns/javaee"  
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
         http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">  
    <servlet>  
        <servlet-name>ABC</servlet-name>  
        <jsp-file>/process.jsp</jsp-file>  
        <init-param>  
            <param-name>driver</param-name>  
            <param-value>oracle.jdbc.driver.OracleDriver</param-value>  
        </init-param>  
    </servlet>  
    <servlet-mapping>  
        <servlet-name>ABC</servlet-name>  
        <url-pattern>/test</url-pattern>  
    </servlet-mapping>  
<welcome-file-list>
```

```
<welcome-file>index.html</welcome-file>
</welcome-file-list>
</web-app>
```

request url

http://localhost:2525/JspApp11/

application object

In jsp, application is an implicit object of type ServletContext.

This instance of ServletContext is created only once by the web container.

This object is used to read initialized parameters from configuration file web.xml file.

Deployment Directory structure

```
JspApp12
|
|---Java Resources
|
|---Web Content
    |
    |---index.html
    |
    |---process.jsp
    |
    |-----WEB-INF
        |
        |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

index.html

```
<center>
    <h1>
        <a href="test"> clickMe </a>
    </h1>
</center>
```

process.jsp

```
<%
    String value=application.getInitParameter("driver");
%>
<center>
    <%= value %>
</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
  <servlet>
    <servlet-name>ABC</servlet-name>
    <jsp-file>/process.jsp</jsp-file>
  </servlet>
  <servlet-mapping>
    <servlet-name>ABC</servlet-name>
    <url-pattern>/test</url-pattern>
  </servlet-mapping>
  <context-param>
    <param-name>driver</param-name>
    <param-value>oracle.jdbc.driver.OracleDriver</param-value>
  </context-param>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
</web-app>
```

request url

http://localhost:2525/JspApp12/

session object

In JSP, session is an implicit object of type HttpSession.

It is used to get or set the session formation.

Deployment Directory structure

```
JspApp13
|
|---Java Resources
|
|---Web Content
  |
  |---form.html
  |
  |---first.jsp
  |
  |---second.jsp
  |
  |-----WEB-INF
```

```
|  
|---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="first.jsp">  
    Name: <input type="text" name="t1"/> <br>  
    <input type="submit" value="submit"/>  
</form>
```

first.jsp

```
<%  
    String name=request.getParameter("t1");  
    out.println("Hello :" + name);  
    session.setAttribute("pname", name);  
%>  
<center>  
    <a href="second.jsp"> click to move for second.jsp</a>  
</center>
```

second.jsp

```
<%  
    String name=(String)session.getAttribute("pname");  
    out.println("Welcome :" + name);  
%>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
         xmlns="http://java.sun.com/xml/ns/javaee"  
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
                           http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">  
  
    <welcome-file-list>  
        <welcome-file>form.html</welcome-file>  
    </welcome-file-list>  
  
</web-app>
```

Request url

http://localhost:2525/JspApp13/

pageContext object

In jsp, pageContext is an implicit object of type pageContext class.

The pageContext object can be used to set, get, remove attributes from one the following scopes., page, request, session, application

In JSP, page scope is a default scope.

Deployment Directory structure

JspApp14

|

|---Java Resources

|

|---Web Content

|

|---form.html

|

|---first.jsp

|

|---second.jsp

|

|-----WEB-INF

|

|----web.xml

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="first.jsp">
    Name: <input type="text" name="t1"/> <br>
    <input type="submit" value="submit"/>
</form>
```

first.jsp

```
<%
    String name=request.getParameter("t1");
    out.println("Hello :" +name);
    pageContext.setAttribute("pname", name, pageContext.SESSION_SCOPE);
%>
<center>
    <a href="second.jsp"> click to move for second.jsp</a>
</center>
```

second.jsp

```
<%
    String
name=(String)pageContext.getAttribute("pname",pageContext.SESSION_SCOPE);
    out.println("Welcome Bro :" + name);
%>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xmlns="http://java.sun.com/xml/ns/javaee"
                     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                     http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

    <welcome-file-list>
        <welcome-file>form.html</welcome-file>
    </welcome-file-list>

</web-app>
```

request url

http://localhost:2525/JspApp14

Junit

Junit is a testing framework which is used to perform unit testing.

Unit testing is a process to test small piece of code working as per the requirement or not.

As a part of TDD(Test Driven Development) , It is highly recommended to perform unit testing by the web developers.

To perform unit testing we need to create test cases or test suit.

Steps to perform unit testing

step1:

Launch eclipse IDE by choosing workspace location.

step2:

Create a java project i.e JunitProj.

ex:

File --> new --> project --> java project --> Next -->
project Name : JunitProj --> Next --> Finish.

step3:

Create "com.ihub.www" package inside "src" folder.

ex:

Right click to src folder --> new --> package -->
Name : com.ihub.www --> finish.

step4:

Create "DemoApp.java" file inside "com.ihub.www" package.

ex:

right click to com.ihub.www package --> new --> class -->
Name: DemoApp --> Finish.

DemoApp.java

```
package com.ihub.www;
public class DemoApp
{
    public String concatenation(String str1, String str2)
    {
        return str1+str2;
    }

    public int sum(int a, int b)
    {
        return a+b;
    }
}
```

step5:

Create a Junit test cases for java methods.

ex:

right click to DemoApp.java file --> new --> others -->
Junit --> Junit Test case --> Next --> select the methods
to whom we want to apply the test cases --> Finish.

step6:

Write the logic to test the test cases inside "DemoAppTest.java" file.

ex:

DemoAppTest.java

```
package com.ihub.www;
import junit.framework.TestCase;
public class DemoAppTest extends TestCase {
    public void testConcatenation() {
        DemoApp da=new DemoApp();
        String result=da.concatenation("ihub", "talent");
        assertEquals("ihubtalent",result );
    }
    public void testSum() {
        DemoApp da=new DemoApp();
        int result=da.sum(10, 20);
        assertEquals(30, result);
    }
}
```

step7:

Run the Junit test cases.

ex:

Right click to the DemoAppTest.java --> run as -->
Junit test case.

Note:

Green color indicates test case is passed.

Brown color indicates test case is failed.

Maven

Maven is a project building management tool.

Maven contains POM.xml file.

Here POM stands for Project Object Model.

A pom.xml file contains dependencies , goals , descriptors and etc.

Steps to develop Maven project

step1:

Launch eclipse IDE by choosing workspace location.

step2:

create a dynamic web project.

ex:

File --> new --> dynamic project --> Name : MavenProj
---> Next --> Next --> generate web.xml file --> finish.

step3:

Convert dynamic web project to Maven project.

ex:

Right click to dynamic project --> configure -->
convert to maven project -->
Group ID : com.ihub.www
Artifact ID : MavenProj
Name : MavenProj
Description: Demostration on Maven project --> finish.

step4:

Create a "ABC.jsp" file inside "Web Content" folder.

ex:

ABC.jsp
<center>
 <h1>
 This is Maven Project Demo
 </h1>
</center>

step5:

Add "servlet-api.jar" file manven depedency inside pom.xml file.

ex:

pom.xml

```
-  
-  
  
<dependencies>  
    <dependency>  
        <groupId>javax.servlet</groupId>  
        <artifactId> servlet-api</artifactId>  
        <version>2.5</version>  
        <scope>provided</scope>  
    </dependency>  
</dependencies>  
<build>  
--  
--  
--
```

step6:

Run the maven project.

ex:

Right click to MavenProject --> run as --> run on server.

step7:

Test the application by using below request url.

ex:

<http://localhost:2525/MavenProj/ABC.jsp>

How to convert Maven project or Dynamic project to war file

step1:

Make sure Dynamic or Maven project is ready in eclipse IDE.

step2:

convert Dynamic or Maven project to war file .

ex:

Right click to MavenProj --> export --> war file -->
Destination : Desktop(choose) --> open --> finish.

GIT/GITHUB

Q) Difference between GIT vs GITHUB ?

GIT

It is distributed version control system
to track changes of each file in a project.
It contains local repository.
It is command line tool.
It is locally installed.

GITHUB

It is a web-based hosting
service for git.
It contains remote repository.
It is GUI.
It is hosted on web.

Q) Types of stages of Git?

We have three stages in git.

Working Directory:

the file exists, but is not part of git's version control.

staging area:

the file has been added to git's version control but changes have not been committed

Repository:

the change has been committed

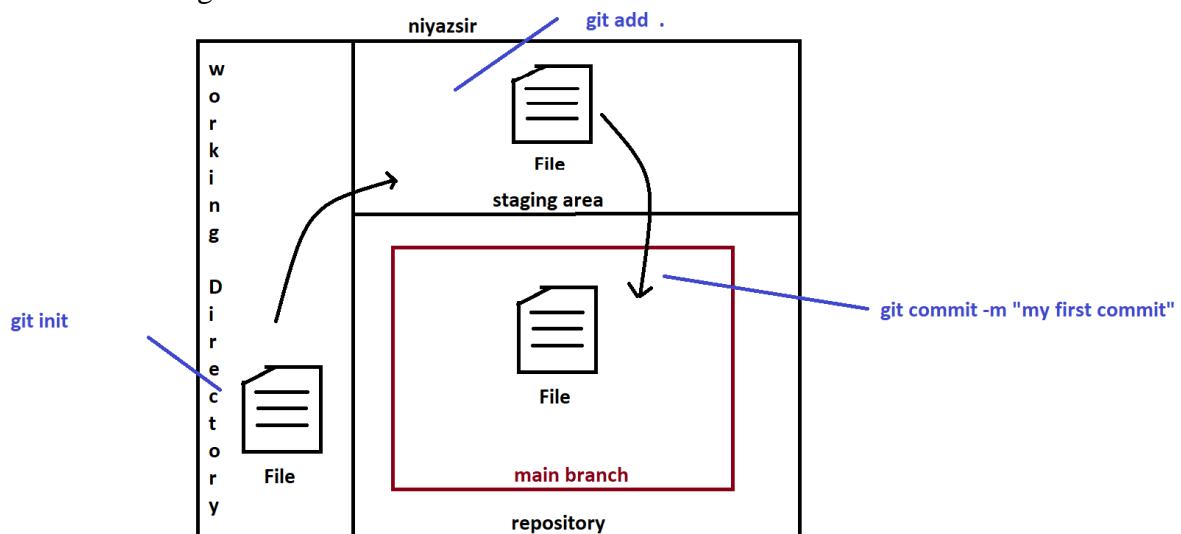


Diagram: git

Remote repository github

<https://github.com/NiyazulHasan/practice>

Q) Write a git command to initialized empty repository?

git init

Q) Write a git command to check the status?

git status

Q) Write a git command to check the branch?

git branch

Q) Write a git command to add remote repository?

git remote add origin <https://github.com/NiyazulHasan/practice>

Q) Write a git command to check the remote repository

git remote -v

Q) Write a git command to commit the changes?

git commit -m "comment here"

Q) Write a git command to push the code to remote origin?

```
git push -f origin main
```

Q) Write a git command to clone the project?

```
git clone <url>
```

Q) Write a git command to pull request?

```
git pull <url>
```

Q) Write a git command to move from master branch to main branch?

```
git branch --move master main
```

SPRING BOOT

Limitations with Spring Framework

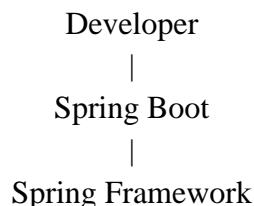
In spring framework a programmer is responsible for following things.

- Adding the dependencies or jar files

- Performing the configurations (applicationContext.xml)

- Managing the physical servers like Tomcat and etc.

- Arranging the physical database like Oracle,MySQL and etc.



Spring Boot

It is an open source java based application framework developed by Pivotal Team.

It provides RAD (Rapid Application Development) features to spring based applications.

It is a production ready grade spring applications with minimum configurations.

In short , **spring boot is a combination of**

Spring Framework + Embedded database + Embedded server

Spring boot does not support xml configuration instead we will use annotations.

Advantages of Spring Boot

It is used to create standalone application which can be started by using java -jar.

It provide production ready grade features like metrix, healthcheck, externalized configurations and etc.

It provides optionate starters to simplify the Maven configurations.

It does not support xml configurations.

It test the web application by using HTTP servers like Tomcat, Jetty or Undertow.

It provides CLI tool for testing and developing spring boot applications.

Interview Questions

Q) What is Spring Boot?

It is an open source java based application framework developed by pivotal team. It provides RAD features to spring based applications. It is a production ready grade spring applications with miniconfigurations.

Q) How many components are there in Spring Boot?

We have four components in spring boot.

- 1) AutoConfiguration
- 2) Starters
- 3) CLI tool
- 4) Actuators

Q) Where we will do configurations in spring boot?

There are two ways to do configurations in spring boot.

- 1) application.properties (default)
- 2) application.yml

Q) List out some embedded servers present in spring boot?

We have following list of embedded servers present in spring boot.

- 1) Tomcat
- 2) Jetty
- 3) Undertow

Q) List out some embedded databases present in spring boot?

We have following list of embedded databases present in spring boot.

- 1) H2
- 2) HSQL
- 3) Derby

Q) In how many ways we can create spring boot application?

There are two ways to create spring boot applications.

- 1) Using Spring Initializr
- 2) Using STS or IntelliJ IDE.

Q) Explain @SpringBootApplication Annotation?

This annotation is a combination of three annotations.

- 1) `@EnableAutoConfiguration` :
It enables Spring Boot's auto-configuration mechanism.
- 2) `@ComponentScan` :
It scans on the package where the application is located.
- 3) `@Configuration` :
It allows us to register extra beans in the context.

Spring Initializr

It is a web based tool which is used to generate spring boot project structure.

ex: <https://start.spring.io/>

Steps to develop first spring boot application

step1:

Goto spring initializr

ex:

<https://start.spring.io/>

step2:

Create a spring boot project i.e FirstSB.

ex:

project : Maven

Language : Java

Dependencies : (don't add)

Spring Boot : 3.1.1

Project Metadata

Group : com.ihub.www

Artifact : SBApp1

Name : SBApp1

Description : Demostration on Spring Boot

Package name: com.ihub.www

packaging : jar

Java : 8

--> click on Generate button.

step3:

Download and Install STS IDE.

STS is a eclipse Based Environment.

ex: <https://spring.io/tools>

step4:

Launch STS IDE by choosing workspace location.

step5:

Extract "SBApp1.zip" file in any loction.

step6:

Open "SBApp1" project in STS IDE.

ex:

File --> Open project from file system --> click to directory button.

--> select FirstSB folder --> Finish.

step7:

Add custom message in SApp1Application.java file.

step8:

Run the spring boot application.

ex: right click to the project --> run as --> spring boot application.

Spring Boot Starters

Spring Boot provides a number of starters that allow us to add jars in the classpath.

Spring Boot built-in starters make development easier and rapid.

Spring Boot Starters are the dependency descriptors.

In the Spring Boot Framework, all the starters follow a similar naming pattern:

spring-boot-starter-*, where * denotes a particular type of application.

ex:

```
spring-boot-starter-test  
spring-boot-starter-web  
spring-boot-starter-validation (bean validation)  
spring-boot-starter-security  
spring-boot-starter-data-jpa  
spring-boot-starter-data-mongodb  
spring-boot-starter-mail
```

Third-Party Starters

We can also include third party starters in our project.

The third-party starter starts with the name of the project.

ex:

```
abc-spring-boot-starter.
```

Spring Boot Starter Web

There are two important features of spring-boot-starter-web.

It is compatible for web development

AutoConfiguration

If we want to develop a web application,we need to add the following dependency in pom.xml file.

ex:

```
<dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-web</artifactId>  
    <version>2.2.2.RELEASE</version>  
</dependency>
```

Spring web starter uses Spring MVC, REST and Tomcat as a default embedded server.

The single spring-boot-starter-web dependency transitively pulls in all dependencies related to web development.

By default, the spring-boot-starter-web contains the following tomcat server dependency:

ex:

```
<dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-tomcat</artifactId>  
    <version>2.0.0.RELEASE</version>  
    <scope>compile</scope>  
</dependency>
```

The spring-boot-starter-web ,auto-configures the following things that are required for the web development:

- 1) Dispatcher Servlet
- 2) Error Page
- 3) Web JARs for managing the static dependencies
- 4) Embedded servlet container

Spring Boot + JSP Application

Project structure

```
SBApp2
|
|---src/main/java
|   |
|   |---com.ihub.www (base package)
|   |   |
|   |   |---SBApp2Application.java
|   |   |---HomeController.java
|---src/main/resources
|   |
|   |---application.properties
|
|---src/test/java
|   |
|   |---SpringBootApp3ApplicationTests.java
|
|--
|--
|--
|---src
|   |
|   |---main
|   |   |
|   |   |---webapp
|   |   |   |
|   |   |   |---pages
|   |   |   |   |
|   |   |   |   |---index.jsp
|---pom.xml
|
```

step1:

Create a spring starter project.

ex:

```
File --> new --> spring starter project -->
          Name : SpringBootApp3
          Group: com.ihub.www
          Artifact: SpringBootApp3
          Description: This is Spring Boot Application with JSP
          package : com.ihub.www ---> next -->
          Starter: Spring Web --> next --> Finish.
```

step2:

create a HomeController class inside "src/main/java".

ex:

```
Right click to package(com.ihub.www) --> new --> class -->
Class: HomeController --> finish.
```

step3:

Add @Controller annotation and "@RequestMapping" annotation inside HomeController class.

HomeController.java

```
package com.ihub.www;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
```

```
@Controller
```

```
public class HomeController {
```

```
    @RequestMapping("/home")
    public String home()
    {
        return "index";
    }
}
```

step4:

create a "webapp" and "pages" folder inside "src/main" folder for adding JSP files.

ex:

```
-----src
  |
  |---main
    |
    |---webapp
      |
      |----pages
```

step5:

create "index.jsp" file inside "src/main/webapp/pages/" folder.

ex:

Right click to pages folder--> new --> file --->
File Name: index.jsp --> finish.

index.jsp

```
<center>
<h1>
    I love Spring Boot Programming
</h1>
</center>
```

step6:

Add "Tomcat Embed Jasper" dependency to read the jsp file.

ex:

```
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
</dependency>
```

Note:

Embedded Tomcat server does not have Jasper. So we need to add above dependency.

step7:

Configure tomcat server port number and jsp file.

application.properties

server.port=9090

```
spring.mvc.view.prefix=/pages/
spring.mvc.view.suffix=.jsp
```

step8:

Run Spring Boot application.

ex:

Right click to MVCApp2 --> run as --> spring boot application.

step9:

Test the application with below request url.

ex:

<http://localhost:9090/home>

Note:-

If you are not getting output please add tomcat dependency in pom.xml file seperately.

ex:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
```

```
<version>3.1.1</version>
</dependency>
```

Interview question

Q) To create a spring mvc based web application we need to add which starter?

spring-boot-starter-web

Q) In spring boot mvc based web application who will pass HTTP request to controller?

DispatcherServlet

Q) Tomcat embedded server by default runs under which port no?

8080

Q) How to change tomcat embedded server port no?

application.properties:- server.port = 9090

Spring Data JPA

Spring Data JPA handles most of the complexity of JDBC-based database access and ORM (Object Relational Mapping).

It reduces the boilerplate code required by JPA(Java Persistence API).

It makes the implementation of your persistence layer easier and faster.

Spring Data JPA aims to improve the implementation of data access layers by reducing the effort to the amount that is needed.

Spring Boot provides spring-boot-starter-data-jpa dependency to connect Spring application with relational database efficiently.

ex:

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
<version>2.2.2.RELEASE</version>
</dependency>
```

The spring-boot-starter-data-jpa internally uses the spring-boot-jpa dependency.

Spring Data JPA provides three repositories are as follows:

CrudRepository:

It offers standard create, read, update, and delete It contains method like findOne(), findAll(), save(), delete(), etc.

PagingAndSortingRepository:

It extends the CrudRepository and adds the findAll methods. It allows us to sort and retrieve the data in a paginated way.

JpaRepository:

It is a JPA specific repository. It is defined in Spring Data Jpa. It extends both the CrudRepository and PagingAndSortingRepository. It adds the JPA-specific methods, like flush() to trigger a flush on the persistence context.

Spring Boot application to interact with H2 Database

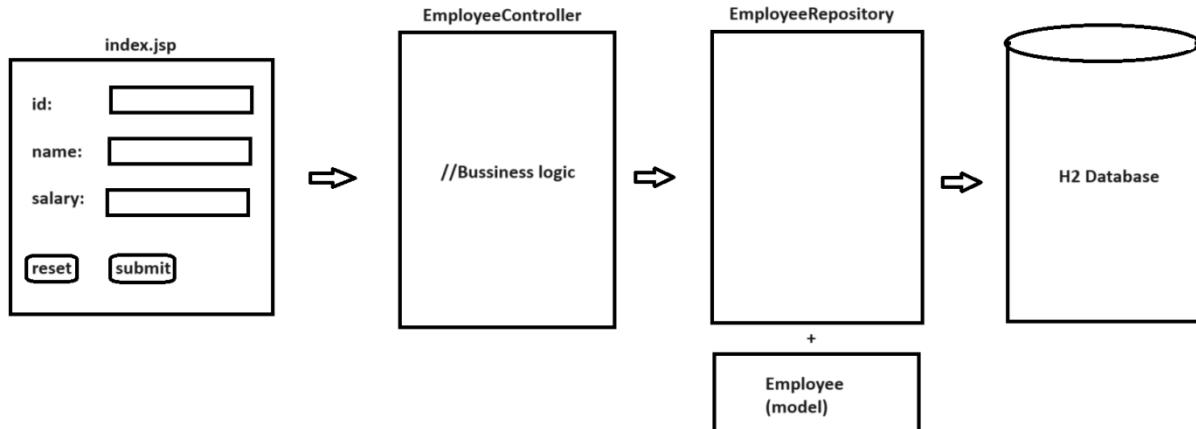


Diagram: sb3.1

project structure

```
SBApp3
|
|----src/main/java
|   |
|   |---com.ihub.www
|   |   |
|   |   |---SBApp3Application.java
|   |
|   |---com.ihub.www.controller
|   |   |
|   |   |---EmployeeController.java (Class)
|   |
|   |---com.ihub.www.repository
|   |   |
|   |   |---EmployeeRepository.java (Interface)
|   |
|   |---com.ihub.www.model
|   |   |
|   |   |---Employee.java (Class)
|
|
|----src/main/resources
|   |
|   |---application.properties
```

```
|----src  
|  
|----main  
|  
|----webapp  
|  
|----index.jsp
```

step1:

Create a spring boot starter project i.e MVCApp2.

ex:

starters:

```
spring web  
spring data jpa  
H2 Database
```

step2:

Add "Tomcat Embed Jasper" dependency to read the jsp file inside pom.xml.

ex:

```
<dependency>  
    <groupId>org.apache.tomcat.embed</groupId>  
    <artifactId>tomcat-embed-jasper</artifactId>  
</dependency>
```

step3:

Create a EmployeeController inside "com.ihub.www.controller" package.

EmployeeController.java

```
package com.ihub.www.controller;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.RequestMapping;  
import com.ihub.www.model.Employee;  
import com.ihub.www.repository.EmployeeRepository;
```

@Controller

```
public class EmployeeController  
{  
    @Autowired  
    EmployeeRepository employeeRepository;  
  
    @RequestMapping("/")  
    public String home()  
    {  
        return "index.jsp";  
    }
```

```

    @RequestMapping("/addEmp")
    public String addEmployeeDetails(Employee e)
    {
        employeeRepository.save(e);

        return "index.jsp";
    }
}

```

step4:

Create index.js file inside "src/main/webapp" folder.

index.js

```

<form action="addEmp">
    <table align="center">
        <caption>Enter the Details</caption>
        <tr>
            <td>Employee Id </td>
            <td><input type="text" name="empId"/></td>
        </tr>
        <tr>
            <td>Employee Name </td>
            <td><input type="text" name="empName"/></td>
        </tr>
        <tr>
            <td>Employee Salary </td>
            <td><input type="text" name="empSal"/></td>
        </tr>
        <tr>
            <td><input type="reset" value="reset"/></td>
            <td><input type="submit" value="submit"/></td>
        </tr>
    </table>
</form>

```

step5:

Create a Employee.java file inside "com.ihub.www.model" package.

Employee.java

```
package com.ihub.www.model;
```

```

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

```

```

@Entity
@Table

```

```

public class Employee
{
    @Id
    private int empId;
    @Column
    private String empName;
    @Column
    private double empSal;

    public int getEmpId() {
        return empId;
    }
    public void setEmpId(int empId) {
        this.empId = empId;
    }
    public String getEmpName() {
        return empName;
    }
    public void setEmpName(String empName) {
        this.empName = empName;
    }
    public double getEmpSal() {
        return empSal;
    }
    public void setEmpSal(double empSal) {
        this.empSal = empSal;
    }
}

```

step6:

Create a EmployeeRepository.java interface inside "com.ihub.www.repository" package.

EmployeeRepository.java

```
package com.ihub.www.repository;
```

```
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;
```

```
import com.ihub.www.model.Employee;
```

```
@Repository
```

```
public interface EmployeeRepository extends CrudRepository<Employee, Integer>
{
```

```
}
```

step7:

Configure server port and h2 database properties inside application.properties file.

application.properties

server.port=9090

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.h2.console.enabled=true

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.hibernate.ddl-auto=update

step8: Run the spring boot starter project.

step9: Test the application by using below request url.

ex: http://localhost:9090

http://localhost:9090/h2-console

RestController:

RestController is used for making restful web services with the help of the @RestController annotation.

This annotation is used at the class level and allows the class to handle the requests made by the client.

The main difference between the @RestController and the @Controller is that the @RestController is a combination of the @Controller and @ResponseBody annotation.

We have following HTTP methods along with rest annotations.

<u>HTTP Methods</u>	<u>Annotations</u>
GET	@GetMapping
POST	@PostMapping
PUT	@PutMapping
Delete	@DeleteMapping
and etc.	

Spring Boot Application using @RestController

Project structure

RestApp

|

|----src/main/java

|

|----com.ihub.www

|

|--RestAppApplication.java

```
|           |--HomeController.java  
|---src/main/resources  
|       |  
|       |----application.properties  
|  
|---src/test/java  
|       |  
|       |----RestAppApplicationTests.java  
|  
|--  
|--  
|--
```

```
|---pom.xml
```

step1: Create a spring starter project.

ex:

```
File --> new --> spring starter project -->  
      Name : RestApp  
      Group: com.ihub.www  
      Artifact: RestApp  
      Description: This is Spring Boot Application  
      package : com.ihub.www --> next -->  
      Starter: Spring Web --> next --> Finish.
```

step2: create a HomeController class inside "src/main/java".

ex:

```
Right click to package(com.ihub.www) --> new -->  
class --> Class: HomeController -->finish.
```

step3: Add @Controller annotation and "@RequestMapping" annotation inside HomeController class.

HomeController.java

```
package com.ihub.www;
```

```
import org.springframework.stereotype.RestController;  
import org.springframework.web.bind.annotation.RequestMapping;
```

```
@RestController  
public class HomeController {
```

```
    @GetMapping("/")  
    public String home()  
    {  
        return "Rest Controll Example";
```

```
}
```

step4: create "index.jsp" file inside "src/main/webapp/pages/" folder.
ex:

Right click to pages folder--> new --> file --->
File Name: index.jsp --> finish.

step5: Configure tomcat server port number and jsp file.

application.properties

server.port=9191

step6: Run Spring Boot application.

ex:

Right click to RestApp --> run as --> spring boot application.

step7: Test the application with below request url.

ex:

<http://localhost:9191/>

Q) Difference between Monolithic Architecture vs Microservice Architecture?

Monolithic Architecture

Monolith means composed all in one piece.

The Monolithic application describes a single-tiered software application in which different components combined into a single program from a single platform.

In Monolithic Architecture we are developing every service individually and at end of the development we are packaging all services as single war file and deploying in a server.

Let's take an example of E-commerce website where we have basic and common option of Customer

Service, Product Service and Cart Service which a customer can access through browser. When we launch the application It is deployed as a single monolithic application. It means we will have only one single instance.

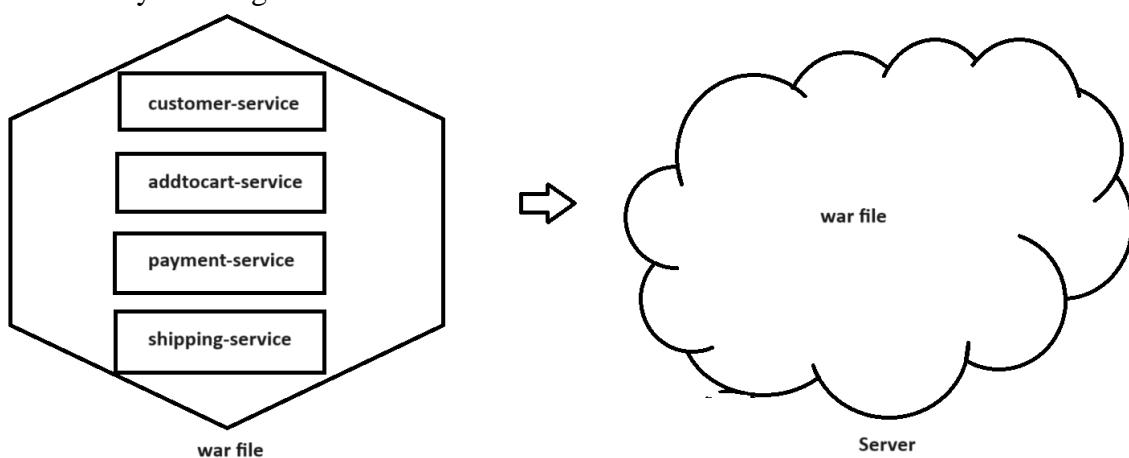


Diagram: sb4.1

Advantages

1) Simple to develop

At the beginning of a project it is much easier to go with Monolithic Architecture.

2) Simple to test

We can implement end-to-end testing by simply launching the application and testing the UI with Selenium.

3) Simple to deploy

we have to copy the packaged application(war file) to a server.

4) Simple to scale

Simple to scale horizontally by running multiple copies behind a load balancer.

Drawbacks of Monolithic Architecture

1) Large and Complex Application

It increase the size of the applications.

It become complex to understand and modify such applications.

As result development slows down and modularity breaks down over the time.

Moreover it is difficult to understand how to currently implement the change due to that quality of code will decline over the time.

2) Slow Development

As application and respective teams grows. The application really become difficult to understand and modify.

Due to large base code which leads to slower the IDE which makes programmers less productive.

3) Blocks Continenous development

In order to update one component/service. we need to re-deploy the entire application which interrupts the background.

There is also a chance ,the components which never have been updated failed to start correctly. As result risk associated with redeployment increases which discourage the continueous development.

4) Unscalable

We can't create instances for a perticular service.

We need to create instance for entire services present in a monolithic application.

5) Unreliable

Every service/component in monolethic application is tightly coupled.

If any one of the service/component goes down the entire system failed to run.

Moreever,A bug in any component/service potentially bring down entire process.

6)Inflexible

It is very difficult to adopt new frameworks and languages.

ex:

Microservices can't communicate eachother. If they written in different languages.

MicroService Architecture

Microservices are the small services that work together

The microservice defines an approach to the architecture that divides an application into a pool of loosely coupled services that implements business requirements.

In Microservice architecture, Each service is self contained and implements a single business capability.

The microservice architectural style is an approach to develop a single application as a suite of small services. It is next to Service-Oriented Architecture (SOA).

Each microservice runs its process and communicates with lightweight mechanisms.

These services are built around business capabilities and independently developed by fully automated deployment machinery.

Advantages of Microservice Architecture

1) Independent Development

Each microservice can be developed independently.

A single development team can build test and deploy the service.

2) Independent Deployment

We can update the service without redeploying the entire application.

Bug release is more manageable and less risky.

3) Fault Tolerance

If service goes down, It won't take entire application down with it.

4) Mixed Technology Stack

It is used to pick best technology which best suitable for our application.

5) Granular Scaling

In Granular scaling, services can scale independently Instead of entire application.

Customer micro service

To develop any micro service we need to follow spring boot flow layered architecture.

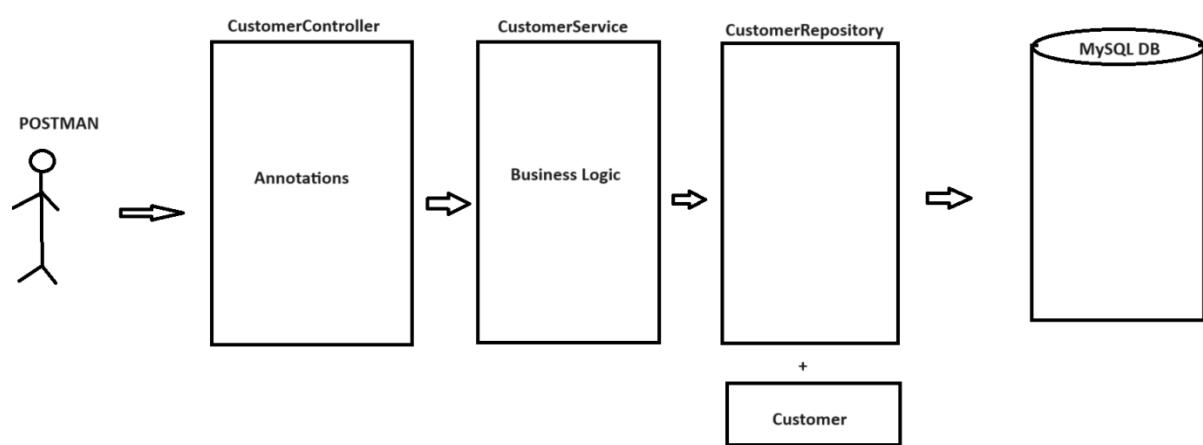


Diagram: sb5.1

step1:

Create a "customer-service" project.

starters:

- spring reactive web
- spring Data JPA
- Project Lombok
- mysql driver

step2:

Create a "demo" schema inside mysql database.

ex:

```
MySQL> create schema demo;  
MySQL> use demo;
```

Project structure

customer-service

|

|----src/main/java

|

|----com.ihub

| |

|----CustomerMicroserviceApplication

|

|----com.ihub.controller

| |

|----CustomerController.java (controller class)

|

|----com.ihub.entity

| |

|----Customer.java (POJO class)

|

|----com.ihub.service

| |

|----CustomerService.java (service class)

|

|----com.ihub.repository

| |

|----CustomerRepository.java (interface)

|

|-----src/main/resources

|

|----application.yml

|

```
|  
|-----pom.xml  
|
```

step3:

Implement the logic as per the requirement.

Customer.java

```
package com.ihub.entity;
```

```
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.Id;  
import javax.persistence.Table;
```

```
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;
```

```
@Entity  
@Data  
@AllArgsConstructor  
@NoArgsConstructor  
public class Customer {
```

```
    @Id  
    @Column(length = 6)  
    private int custId;
```

```
    @Column(length = 12)  
    private String custName;
```

```
    @Column(length = 12)  
    private String custAddress;
```

```
}
```

CustomerRepository.java

```
package com.ihub.repository;
```

```
import org.springframework.data.jpa.repository.JpaRepository;
```

```
import com.ihub.entity.Customer;
```

```
public interface CustomerRepository extends JpaRepository<Customer, Integer> {  
}
```

CustomerService.java

```
package com.ihub.www.service;  
  
import java.util.List;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.data.repository.config.CustomRepositoryImplementationDetector;  
import org.springframework.stereotype.Service;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
  
import com.ihub.www.model.Customer;  
import com.ihub.www.repo.CustomerRepository;  
  
@Service  
public class CustomerService  
{  
    @Autowired  
    CustomerRepository customerRepository;  
  
    public Customer addCustomer(Customer customer)  
    {  
        return customerRepository.save(customer);  
    }  
  
    public List<Customer> getAllCustomers()  
    {  
        return customerRepository.findAll();  
    }  
  
    public Customer getCustomerById(int custId)  
    {  
        return customerRepository.findById(custId).get();  
    }
```

```

public String updateCustomer(Customer customer)
{
    Customer cust=customerRepository.findById(customer.getCustId()).get();

    cust.setCustName(customer.getCustName());
    cust.setCustAdd(customer.getCustAdd());

    customerRepository.save(cust);

    return "Record updated";
}

public String deleteCustomer(int custId)
{
    Customer cust=customerRepository.findById(custId).get();
    customerRepository.delete(cust);

    return "Record Deleted";
}
}

```

CustomerController.java

```

package com.ihub.www.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.service.annotation.PutExchange;

import com.ihub.www.model.Customer;
import com.ihub.www.service.CustomerService;

@RestController
@RequestMapping("/customer")
public class CustomerController
{

```

```

@Autowired
CustomerService customerService;

@PostMapping("/add")
public Customer addCustomer(@RequestBody Customer customer)
{
    return customerService.addCustomer(customer);
}

@GetMapping("/fetch")
public List<Customer> getAllCustomers()
{
    return customerService.getAllCustomers();
}

@GetMapping("/fetch/{custId}")
public Customer getCustomerById(@PathVariable int custId)
{
    return customerService.getCustomerById(custId);
}

@PutMapping("/update")
public String updateCustomer(@RequestBody Customer customer)
{
    return customerService.updateCustomer(customer);
}

@DeleteMapping("/delete/{custId}")
public String deleteCustomer(@PathVariable int custId)
{
    return customerService.deleteCustomer(custId);
}
}

```

application.yml

```

server:
  port: 9090

```

```

spring:
  application:
    name: CUSTOMER-SERVICE

```

```

datasource:

```

```

driver-class-name: com.mysql.jdbc.Driver
url: jdbc:mysql://localhost:3306/demo
username: root
password: root
jpa:
  hibernate.ddl-auto: update
  generate-ddl: true
  show-sql: true

```

step3:

Run the spring boot application and get check below url.

<u>METHODS</u>	<u>URL</u>
GET	http://localhost:9001/customer/fetch
GET	http://localhost:9001/customer/fetch/101
POST	http://localhost:9001/customer/add
	> body >raw
	{ "custId":101, "custName":"Alex", "custAdd":"Chicago" }
PUT	http://localhost:9001/customer/update
DELETE	http://localhost:9001/customer/delete/101

Exception Handling in Spring Boot

If we give/pass wrong request to our application then we will get Exception.

ex: http://localhost:9090/fetch/102

Here '102' record is not available so immediately our controller will throw below exception.

ex:

```
{
  "timestamp": "2021-02-14T06:24:01.205+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "path": "/fetch/102"
}
```

Handling exceptions and errors in APIs and sending the proper response to the client is good for enterprise applications.

In Spring Boot Exception handling can be performed by using Controller Advice.

@ControllerAdvice

The `@ControllerAdvice` is an annotation used to handle the exceptions globally.

@ExceptionHandler

The `@ExceptionHandler` is an annotation used to handle the specific exceptions and sending the custom responses to the client.

project structure

customer-service

|

|---src/main/java

|

|---com.ihub.www

|

|---CustomerServiceApplication.java

|

|---com.ihub.www.controller

|

|---CustomerController.java

|---com.ihub.www.service

|

|---CustomerService.java

|---com.ihub.www.repo

|

|---CustomerRepository.java(Interface)

|---com.ihub.www.model

|

|---Customer.java

|---com.ihub.www.exception

|

|---ErrorDetails.java(POJO)

|

|---ResourceNotFoundException.java

|

|---GlobalExceptionHandler.java

|-----src/main/resources

|

|---application.properties

|

|---pom.xml

step1: Use the existing project i.e customer-service.

step2: Create a com.ihub.www.exception package inside "src/main/java".

step3: Create ErrorDetails.java file inside "com.ihub.www.exception" pkg.

ErrorDetails.java

```
package com.ihub.www.exception;
import java.util.Date;
public class ErrorDetails
{
    private Date timestamp;
    private String message;
    private String details;

    public ErrorDetails(Date timestamp, String message, String details) {
        super();
        this.timestamp = timestamp;
        this.message = message;
        this.details = details;
    }

    public Date getTimestamp() {
        return timestamp;
    }
    public void setTimestamp(Date timestamp) {
        this.timestamp = timestamp;
    }
    public String getMessage() {
        return message;
    }
    public void setMessage(String message) {
        this.message = message;
    }
    public String getDetails() {
        return details;
    }
    public void setDetails(String details) {
        this.details = details;
    }
}
```

step4: Create ResourceNotFoundException.java file inside "com.ihub.www.exception" pkg.

ResourceNotFoundException.java

```
package com.ihub.www.exception;

public class ResourceNotFoundException extends RuntimeException
{
    public ResourceNotFoundException(String msg)
    {
        super(msg);
    }
}
```

step5: Create a GlobalExceptionHandler.java file inside
"com.ihub.www.exception" pkg.

GlobalExceptionHandler.java

```
package com.ihub.www.exception;

import java.util.Date;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.context.request.WebRequest;

@ControllerAdvice
public class GlobalExceptionHandler
{

    @ExceptionHandler(ResourceNotFoundException.class)
    public ResponseEntity<?> handleResourceNotFoundException
    (ResourceNotFoundException exception,WebRequest request )
    {

        ErrorDetails errorDetails=new ErrorDetails(new
Date(),exception.getMessage(),request.getDescription(false));
        return new ResponseEntity<>(errorDetails,HttpStatus.NOT_FOUND);
    }

    //handle global exception
    @ExceptionHandler(Exception.class)
    public ResponseEntity<?> handleException
```

```

        (Exception exception,WebRequest request )
        {
            ErrorDetails          errorDetails=new           ErrorDetails(new
Date(),exception.getMessage(),request.getDescription(false));
            return               new
ResponseEntity<>(errorDetails,HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }
}

```

step6: Now add ResourceNotFoundException to CustomerService.

CustomerService.java

```

package com.ihub.www.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.ihub.www.exception.ResourceNotFoundException;
import com.ihub.www.model.Customer;
import com.ihub.www.repo.CustomerRepository;

@Service
public class CustomerService
{
    @Autowired
    CustomerRepository customerRepository;

    public Customer addCustomer(Customer customer)
    {
        return customerRepository.save(customer);
    }

    public List<Customer> getAllCustomer()
    {
        return customerRepository.findAll();
    }

    public Customer getCustomer(int custId)
    {
        return customerRepository.findById(custId)
            .orElseThrow(()-> new ResourceNotFoundException("ID NOT
FOUND"));
    }
}

```

```

}

public String updateCustomer(Customer customer)
{
    Customer cust=customerRepository.findById(customer.getCustId()).get();

    cust.setCustName(customer.getCustName());
    cust.setCustAdd(customer.getCustAdd());

    customerRepository.save(cust);

    return "Record updated";
}

public String deleteCustomer(int custId)
{
    Customer customer=customerRepository.findById(custId)
        .orElseThrow(()->new ResourceNotFoundException("Id Not Found for
Delete"));

    customerRepository.delete(customer);

    return "Record is deleted";
}
}

```

step7: Relaunch the spring boot application.

step8: Test the application by using below request url.

ex:

http://localhost:9090/fetch/102

step9: Here exception will display in below format.

ex:

```
{
    "timestamp": "2023-03-27T23:04:03.181+00:00",
    "message": "ID NOT FOUND",
    "details": "uri=/fetch/102"eureka
}
```

Types of API's

PIs are broadly accepted and used in web applications.

There are four different types of APIs commonly used in web services.

1) public API

A public API is open and available for use by any outside developer or business.

2) partner API

A partner API, only available to specifically selected and authorized outside developers or API consumers, is a means to facilitate business-to-business activities.

3) private API / Internal APIs

An internal or private API is intended only for use within the enterprise to connect systems and data within the business.

4) composite API

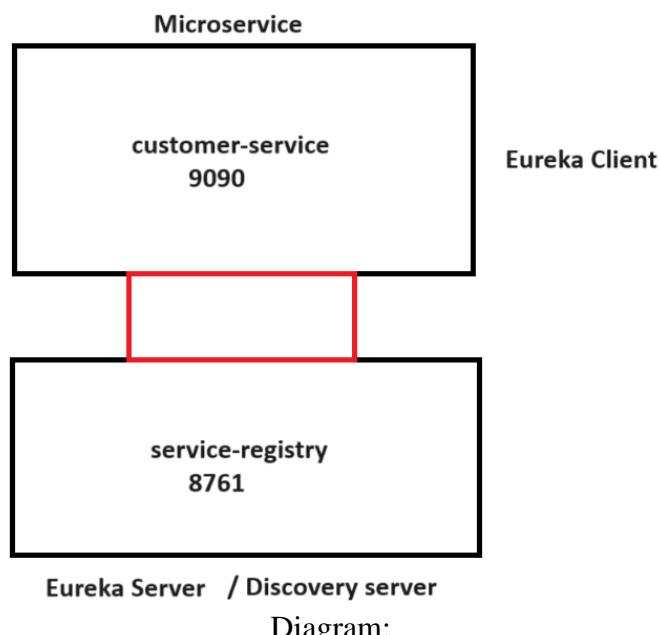
Composite APIs generally combine two or more APIs to craft a sequence of related or interdependent operations.

Eureka Server

This server holds information about the client service applications.

Each microservice registers into Eureka server and eureka server knows all client applications running on each port and IP address.

This server is also known as discovery server.



step1: Add Eureka Client dependency in "customer-service" project.

ex:

starter

Eureka Discovery client.

step2: Create a "service-registry" project to register all microservices.

Here "service-registry" is a Eureka Server and microservices are Eureka Clients.

```
> service-registry  
  starter  
    > Eureka Server.
```

step3: Add "@EnableEurekaServer" annotation in main spring boot application.

ServiceRegisterApplication.java

```
package com.ihub;
```

```
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;
```

```
@SpringBootApplication  
@EnableEurekaServer  
public class ServiceRegisterApplication {  
  
    public static void main(String[] args) {  
        SpringApplication.run(ServiceRegisterApplication.class, args);  
    }  
  
}
```

step4: Add port number and set register for Eureka service as false.

application.yml

```
server:
```

```
  port: 8761
```

```
eureka:
```

```
  client:
```

```
    register-with-eureka: false
```

```
    fetch-registry: false
```

step5: Open the "customer-service" application.yml and add
register with eureka as true.

application.yml

```
server:
```

```
  port: 9001
```

```
spring:  
  application:  
    name: CUSTOMER-SERVICE
```

```
datasource:
```

```

driver-class-name: com.mysql.jdbc.Driver
url: jdbc:mysql://localhost:3306/demo
username: root
password: root
jpa:
  hibernate.ddl-auto: update
  generate-ddl: true
  show-sql: true

eureka:
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://localhost:8761/eureka/
  instance:
    hostname: localhost

```

step6: Now run all two projects.

First run service-registry then customer-service.
 First run eureka server then eureka client.

step7: Check the output in below url's.

ex:
<http://localhost:8761/>

Spring Cloud API Gateway

Spring Cloud Gateway aims to provide a simple, effective way to route to API's and provide cross cutting concerns to them such as security,monitoring/metrics, authentication, authorization ,adaptor and etc.

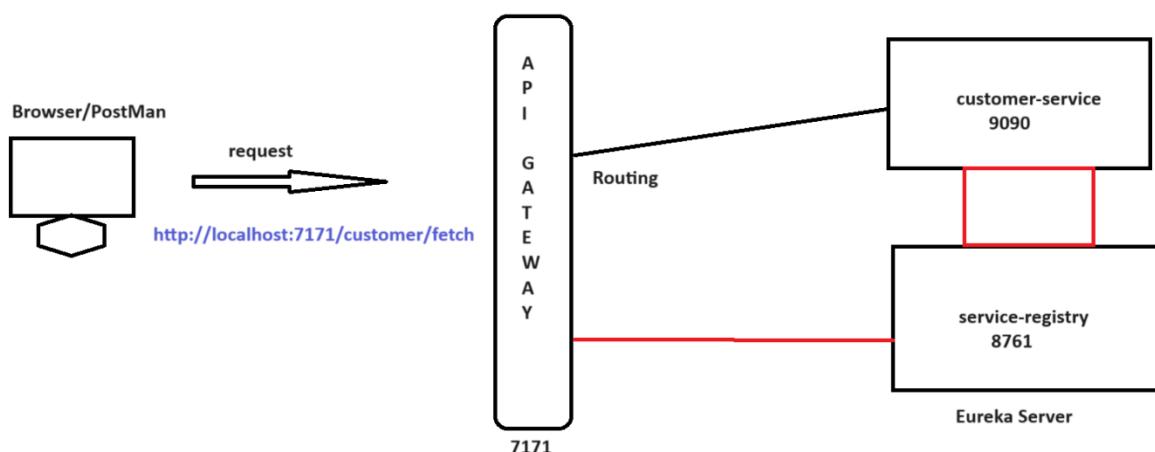


Diagram: sb6.2

step1: Create a "cloud-apigateway" project in STS.

starters:

- eureka Discovery client
- Spring boot actuators
- spring reactive web

step2: Add spring cloud dependency in pom.xml file.

ex:

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-gateway</artifactId>
    <version>3.1.1</version>
</dependency>
```

step3: Add "@EnableEurekaClient" annotation on main spring boot application.

CloudApigatewayApplication.java

```
package com.ge;
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
```

```
@SpringBootApplication
@EnableEurekaClient
public class CloudApigatewayApplication {

    public static void main(String[] args) {
        SpringApplication.run(CloudApigatewayApplication.class, args);
    }

}
```

step4: Register port number, set application name, and configure all microservices for routing in application.yml file.

application.yml

```
server:
  port: 7171
```

```
eureka:
```

```
  client:
```

```
    register-with-eureka: true
    fetch-registry: true
```

```

service-url:
  defaultZone: http://localhost:8761/eureka/
instance:
  hostname: localhost

spring:
  application:
    name: API-GATEWAY
  cloud:
    gateway:
      routes:
        - id: CUSTOMER-SERVICE
          uri: lb://CUSTOMER-SERVICE
          predicates:
            - Path=/customer/**

```

step5: Now Run the following applications sequentially.

- "service-registry"
- "customer-service"
- "cloud-apigateway".

step6: Test the applications by using below urls.

ex:

- http://localhost:9191/customer/fetch/101
- http://localhost:9191/customer/fetch

Spring Cloud Hystrix

Hystrix is a fault tolerance library provided by Netflix.

Using Hystrix we can prevent Deligation of failure from one service to another service.

Hystrix internally follows Circuit Breaker Design pattern.

In short circuit breaker is used to check availability of external services like web service call, database connection and etc.

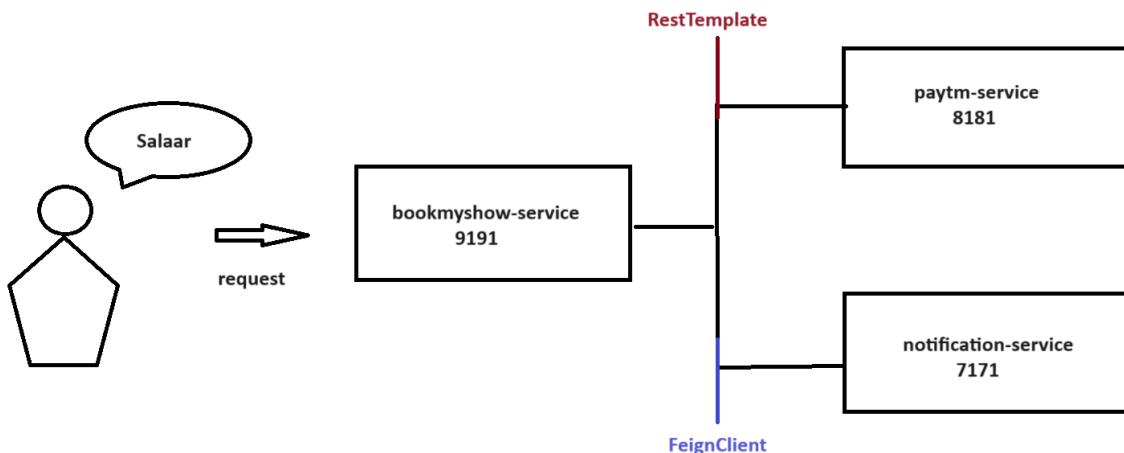


Diagram: sb8.1

notification-service

step1: create a "notification-service" project in STS.

Starter: Spring Web.

step2: Add the following code in main spring boot application.

NotificationServiceApplication.java

```
package com.ihub.www;
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
@RequestMapping("/notification")
public class NotificationServiceApplication {

    @GetMapping("/send")
    public String sendEmail()
    {
        return "Email sending method is called from notification-service";
    }
    public static void main(String[] args) {
        SpringApplication.run(NotificationServiceApplication.class, args);
    }
}
```

step3: convert application.properties file to application.yml file.

step4: configure server port number in application.yml file.

application.yml

```
server:
```

```
  port: 7171
```

step5: Run "notification-service" project as spring boot application.

step6: Test the application with below request url.

ex:

```
http://localhost:7171/notification/send
```

paytm-service

step1: create a "paytm-service" project in STS.

Starter: Spring Web.

step2: Add the following code in main spring boot application.

PaytmServiceApplication.java

```
package com.ihub.www;
```

```
import org.springframework.boot.SpringApplication;
```

```
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
import org.springframework.web.bind.annotation.GetMapping;
```

```
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RestController;
```

```
@SpringBootApplication
```

```
@RestController
```

```
@RequestMapping("/paytm")
```

```
public class PaytmServiceApplication {
```

```
    @GetMapping("/pay")
```

```
    public String paymentProcess()
```

```
{
```

```
        return "Payment Pocess method called in paytm-service";
```

```
}
```

```
    public static void main(String[] args) {
```

```
        SpringApplication.run(PaytmServiceApplication.class, args);
```

```
}
```

```
}
```

step3: convert application.properties file to application.yml file.

step4: configure server port number in application.yml file.

application.yml

```
server:
```

```
    port: 8181
```

step5: Run "paytm-service" project as spring boot application.

step6: Test the application with below request url.

```
ex:
```

```
http://localhost:8181/paytm/pay
```

bookmyshow-service

step1: create a "bookmyshow-service" project in STS.

Starter: Spring Web

step2: Add Spring Cloud Hystrix dependency in pom.xml file.

ex:

```
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
<version>2.2.10.RELEASE</version>
</dependency>
```

step3: Change <parent> tag inside pom.xml file for hystrix compatibility.

ex:

```
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.3.3.RELEASE</version>
<relativePath /> <!-- lookup parent from repository -->
</parent>
```

step4: Add the following code in main spring boot application.

BookmyshowServiceApplication

```
package com.ihub.www;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.hystrix.EnableHystrix;
import org.springframework.context.annotation.Bean;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

import com.netflix.hystrix.contrib.javanica.annotation.HystrixCommand;

@SpringBootApplication
@RestController
@EnableHystrix
public class BookmyshowServiceApplication {

    @Autowired
    RestTemplate restTemplate;
```

```

    @HystrixCommand(groupKey = "ihub" , commandKey = "ihub" ,fallbackMethod =
"bookMyShowFallBack")
    @GetMapping("/book")
    public String bookShow()
    {
        String
paytmServiceResponse=restTemplate.getForObject("http://localhost:8181/paytm/pay",
String.class);
        String
notificationServiceResponse=restTemplate.getForObject("http://localhost:7171/notification/s
end",String.class);

        return paytmServiceResponse+"\n"+notificationServiceResponse;
    }

public static void main(String[] args) {
    SpringApplication.run(BookmyshowServiceApplication.class, args);

}

public String bookMyShowFallBack()
{
    return "service gateway failed";
}

@Bean
public RestTemplate getRestTemplate() {

    return new RestTemplate();

}

}

```

step5: convert application.properties file to application.yml file.
 step6: configure server port number inside application.yml file.

application.yml

```

server:
  port: 9191

```

step7: Add spring core dependency inside pom.xml file.

ex:

```

<dependency>
```

```
<groupId>org.springframework</groupId>
<artifactId>spring-core</artifactId>
<version>5.3.17</version>
</dependency>
```

step8: Run the "bookmyshow-service" application as spring boot application.

step9: Test the application by using below request url.

ex: http://localhost:9191/book

step10: Now stop any micro service i.e notification-service or paytm-service.

step11: Test the "bookmyshow-service" application by using below url.

ex:

http://localhost:9191/book

Note: Here fallback method will execute with the help of Hystrix.

Spring Security

Spring Security is a framework which provides various security features like authentication, authorization to create secure Java Enterprise Applications.

It is a sub-project of Spring framework which was started in 2003 by Ben Alex.

Later on, in 2004, It was released under the Apache License as Spring Security 2.0.0.

This framework targets two major areas of application

1) Authentication

It is a process of knowing and identifying the user that wants to access.

2) Authorization

It is a process to allow authority to perform actions in the application.

Project structure

SpringSecurityApp

|

|---src/main/java

|

|---com.ge.www

|

|---|--SpringSecurityAppApplication.java

|

|---com.ge.www.controller

|

|---|--HomeController.java

|

|---src/main/resources

|

|----application.yml

|

|---src/test/java

|

|----SSpringSecurityApplicationTests.java

```
|---pom.xml  
|  
|
```

step1: create a spring starter project.

starters: spring web
 spring security.

step2: create a Controller to accept the request.

HomeController.java

```
package com.ge.www.controller;  
  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class HomeController {  
  
    @GetMapping("/msg")  
    public String msg()  
    {  
        return "Welcome to Spring Security";  
    }  
}
```

step3: Configure server port number in application.properties file.

application.yml

```
server:  
port: 9191
```

step4: Run the application as spring boot application.

step6: Test the application by using below url.

ex:
http://localhost:9191/msg

Note: When we hit the request ,we will get login page.

Default username is "user" and password we can copy from STS console.

step7: To change the default user and password we can use below properties in application.properties file.

application.yml

```
server:  
  port: 9191
```

```
spring  
  security:  
    user:  
      name=raja  
      password=rani
```

step8: Relaunch the spring boot application.

step9: Test the application by using below url.

ex:

<http://localhost:9191/msg>

How can we convert spring project to jar file

step1: Make sure spring boot project is ready.

step2: Create a jar file for spring boot project.

ex: right click to project --> run as --> Maven build -->
Goals: package --> run.

step3: Check the jar file inside target folder of a spring boot project.

Q) What is the difference between WaterFall Model vs Agile Methodology?

Agile

It processes non-linear,incremental and interactive approach for a software design.

At each incremental approach project is tested.

A customer has early and frequent opportunities to look at the product and make decisions or changes to the project.

It is unstructured because it is not plan oriented.

Small projects can be implemented quickly.But large projects can't be estimated and implemented due to frequent changes.

WaterFall Model

It processes linear and sequential approach for a software design.

At the end whole project is tested.

A customer only can see that product at the end of the project.

It is structured because it is plan oriented.

All sort of projects can be estimated and implemented.

HTML

HTML5

HTML stands for Hypertext Markup language.

It is widely used language on web to develop web pages and web applications.

It is developed by Burners Lee in late 1991.

It is a case insensitive language.

It is a tag based language.

ex: <html>

Every tag contains opening tag and closing tag.

ex: <html> -- opening tag

</html> -- closing tag

Every tag contains attributes and each attribute contains attribute name and attribute value.

ex: <body bgcolor="red">

Tag may contains multiple attributes and each attribute seperated with space.

ex: <body bgcolor="red" background="image">

HTML file is know as HTML document.

All HTML document we need to save with .html or .htm extension.

All HTML documents execute in a browser window.

ex: Chrome

Mozilla

Edge

Opera

IE and etc.

We will use following editors to write HTML code.

ex: Notepad

Notepad++

Sublime

VSC

Dreamviewer and etc.

A simple HTML document is also known as component.

HTML is used to develop client side components.

HTML 2.0 was released in the year of 1995.

HTML 4.0 was released in the year of 1999.

HTML5 which is a extension of HTML4 was released in 2012.

The main objective of HTML5 is used to develop light weight component.

HTML5 is also known as advanced hypertext markup language.

HTML skeleton

HTML

```
<!DOCTYPE>
```

```
<html>
```

```
    <head>
```

```
        - // head related tags
```

```
    -
```

```
    </head>
```

```
    <body>
```

```
        - // body related tags
```

```
    -
```

```
</body>
</html>

HTML5
<!DOCTYPE html>
<html>
    <head>
        -
        - // head related tags
        -
    </head>
    <body>
        -
        - // body related tags
        -
    </body>
</html>
```

Note: <!DOCTYPE> represent HTML document.
<!DOCTYPE html> represent HTML5 document.

<html> A <html> tag is a root tag for entire HTML document.
<head> A <head> tag is used to declare following things.
ex: title of a web page
 favicon of a web page
 styles
 scripts
<body> A <body> tag is used to declare actual content of a web page.
ex: images
 forms
 tables
 lists and etc.

First HTML Document

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        Welcome to HTML classes
    </body>
</html>
```

Q) How can we add title to a web page?

To add title of a web page we need to use <title> tag.
ex: <!DOCTYPE html>
<html>
<head>
 <title>IHUB TALENT</title>
</head>
<body>Welcome to HTML classes</body>
</html>

Q) How to add favicon to a web page?

A <link> tag is used to add favicon on a web page.

ex: <!DOCTYPE html>

```
<html>
  <head>
    <title>IHUB TALENT</title>
    <link rel="icon" href="ihub.jpg">
  </head>
  <body>
    Welcome to HTML classes
  </body>
</html>
```

Q) How to change the background color in html?

```
<!DOCTYPE html>
<html>
  <head>
    <title>IHUB TALENT</title>
    <link rel="icon" href="ihub.jpg">
  </head>
  <body bgcolor="yellow">Welcome to HTML classes</body>
</html>
```

Q) How to set the background image in html?

```
<!DOCTYPE html>
<html>
  <head>
    <title>IHUB TALENT</title>
    <link rel="icon" href="ihub.jpg">
  </head>
  <body background="bg.jpg">
    </body>
</html>
```

HTML Meta Tags

A meta tag is used to declare metadata of a document.

Here metadata means data of a data.

Metadata used by web browser, search engine and other web services.

To declare metadata we need to use <meta> tag.

A <meta> tag we can declare inside <head> tag.

A <meta> tag is used to declare following things.

- 1) Description
- 2) Author
- 3) keywords
- 4) view port
- 5) refresh
- 6) copyright

and etc.

UTF-8

UTF stands for Unicode Transformation Format.

UTF is a encoding methods which describes what character set a website is written with.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <!-- title -->
        <title>MyPage!</title>

        <!-- favicon -->
        <link rel="icon" href="ihub.jpg">

        <!-- meta data -->
        <meta charset="utf-8">
        <meta name="description" content="Web page for learning purpose">
        <meta name="author" content="Niyaz sir">
        <meta name="keywords" content="HTML      CSS      JAVASCRIPT
BOOSTRAP">
        <meta name="viewport" content="width=device-width, intial-scale=1.0">
        <meta http-equiv="refresh" content="05">
        <meta name="copyright" content="Copy right by Niyaz sir">
    </head>
    <body>
        This is HTML class
    </body>
</html>
```

Basic Tags in HTML

Heading Tags

It will display the text in bold and size of the text is depends upon number of heading tags.

We have 6 heading tags from h1 to h6.

ex: <!DOCTYPE html>

```
<html>
    <head>
        <!-- title -->
        <title>MyPage!</title>

        <!-- favicon -->
        <link rel="icon" href="ihub.jpg">
    </head>
    <body>
        <h1> Heading Tag 1</h1>
        <h2> Heading Tag 2</h2>
        <h3> Heading Tag 3</h3>
        <h4> Heading Tag 4</h4>
        <h5> Heading Tag 5</h5>
        <h6> Heading Tag 6</h6>
    </body>
</html>
```

Paragraph tag

A <p> tag is used to display the text in paragraph.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <!-- title -->
        <title>MyPage!</title>

        <!-- favicon -->
        <link rel="icon" href="ihub.jpg">

    </head>
    <body>
        <p>
            HTML is widely used language on web to develop
            web pages and web applications and it is developed by
            burners lee in late 1991.
        </p>
    </body>
</html>
```

center tag

A <center> tag is used to display the text in center.

HTML5 does not support <center> tag.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <!-- title -->
        <title>MyPage!</title>

        <!-- favicon -->
        <link rel="icon" href="ihub.jpg">

    </head>
    <body>
        <center>This is center tag</center>
    </body>
</html>
```

bold tag

A tag is used to display the text in bold without importance.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <!-- title -->
        <title>MyPage!</title>
```

```
<!-- favicon -->
<link rel="icon" href="ihub.jpg">

</head>
<body>
    <b>This is bold tag</b>
</body>
</html>
```

strong tag

A **** tag is used to display the text in bold with important.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <!-- title -->
        <title>MyPage!</title>

        <!-- favicon -->
        <link rel="icon" href="ihub.jpg">
    </head>
    <body>
        <strong>This is strong tag</strong>
    </body>
</html>
```

italic tag

A *<i>* tag is used to display the text in italic without force.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <!-- title -->
        <title>MyPage!</title>

        <!-- favicon -->
        <link rel="icon" href="ihub.jpg">
    </head>
    <body>
        <i>This is italic tag</i>
    </body>
</html>
```

emphasize tag

A ** tag is used to display the text in italic with force.

ex:

```
<!DOCTYPE html>
<html>
    <head>
```

```
<!-- title -->
<title>MyPage!</title>

<!-- favicon -->
<link rel="icon" href="ihub.jpg">

</head>
<body>
    <em>This is emphasize tag</em>
</body>
</html>
```

underline tag

A <u> tag is used to display the text in underline.
HTML5 does not support <u> tag.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <!-- title -->
        <title>MyPage!</title>

        <!-- favicon -->
        <link rel="icon" href="ihub.jpg">
    </head>
    <body>
        <u>This is underline tag</u>
    </body>
</html>
```

font tag

A font tag is used to apply the color to the text , to increase the font size and change font family.

HTML 5 does not support tag.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <!-- title -->
        <title>MyPage!</title>

        <!-- favicon -->
        <link rel="icon" href="ihub.jpg">
    </head>
    <body>
        <font color="blue" size="30" face="cursive">
            This is font tag
        </font>
    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <!-- title -->
        <title>MyPage!</title>

        <!-- favicon -->
        <link rel="icon" href="ihub.jpg">
    </head>
    <body>
        <font color="blue" size="30" face="monospace">
            This is font tag
        </font>
    </body>
</html>
```

breakline tag

A `
` tag is used to break the line in a web page.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <!-- title -->
        <title>MyPage!</title>

        <!-- favicon -->
        <link rel="icon" href="ihub.jpg">

    </head>
    <body>
        <i>This is italic tag</i>
        <br>
        <b>This is bold tag</b>
    </body>
</html>
```

Horizontal line

A `<hr>` tag is used to display horizontal line in a web page.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <!-- title -->
        <title>MyPage!</title>

        <!-- favicon -->
        <link rel="icon" href="ihub.jpg">

    </head>
```

```

<body>
    <i>This is italic tag</i>
    <hr>
    <b>This is bold tag</b>
</body>
</html>

```

marquee tag

A <marquee> tag is used to scroll the text.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <!-- title -->
        <title>MyPage!</title>
        <!-- favicon -->
        <link rel="icon" href="ihub.jpg">
    </head>
    <body>
        <marquee> This is HTML class for Freshers </marquee>
    </body>
</html>

```

Nested Tags in HTML

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <center>
            <h1>
                <font color="blue">Welcome to HTML classes</font>
            </h1>
        </center>
    </body>
</html>

```

Phrase tags in HTML

Phrase tags are special purpose tags which are used to define structural meaning to block of code or text.

We have following list phrase tags in html.

- 1) abbreviation - <abbr>
 - 2) definition - <dfn>
 - 3) short quote - <q>
 - 4) address - <address>
 - 5) code - <code>
 - 6) keyboard - <kbd>
 - 7) strike - <strike> or <s>
- and etc.

1) abbreviation - <abr>

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <abr title="HyperText Markup Language">
      HTML
    </abr>
    is a markup language and it is a case insensitive language.

  </body>
</html>
```

2) definition - <dfn>

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <dfn>HTML</dfn>
    is a markup language and it is a case insensitive language.
  </body>
</html>
```

3) short quote - <q>

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <q>I love html coding because it is easy</q>
  </body>
</html>
```

4) address - <address>

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <address>
      1-4-78/1 , Nilgiri Block, Ameerpet, Hyderabad.
    </address>
  </body>
</html>
```

5) code - <code>

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <code>
      void main()
      {
        clrscr();
        printf("Hello");
        getch();
      }
    </code>
  </body>
</html>
```

6) keyboard - <kbd>

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <p> To copy press CTRL+C button </p>
    <p> To copy press <kbd>CTRL+C</kbd> button </p>
  </body>
</html>
```

7) strike - <strike> or <s>

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <h1>
      <s>
        This is Java Class
      </s>
    </h1>
  </body>
</html>
```

HTML images

A `` tag is used to display the images in a web page.

It is an opening tag with attributes and does not have any closing tag.

A `` tag contains following attributes.

- 1) `src` - It is used to locate a file.
- 2) `alt` - It will display alternate message if image is not found.
- 3) `width` - It is used to set the width to the image.
- 4) `height` - It is used to set the height to the image.

We have following list of images.

<u>Format</u>	<u>Abbreviation</u>
JPEG	Joint Photographic Expert Group
PNG	Portable Network Graphics
SVG	Scalable Vector Graphics
GIF	Graphical Interchange Format and etc.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        
    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        
    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        
    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    
  </body>
</html>
```

Hyperlink in HTML

A [anchor tag](#) is used to display hyperlink in a web page.
Anchor tag contains "href" attribute to navigate to other resources.

ex:1

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <h1>
      <a href="http://www.facebook.com/login"> Facebook </a>
    </h1>
  </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <h1>
      <a href="http://www.facebook.com/login"
         target="_self">
        Facebook
      </a>
    </h1>
  </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <h1>
      <a href="http://www.facebook.com/login"
         target="_blank">
        Facebook
      </a>

    </h1>
  </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <center>
      <a href="https://en.wikipedia.org/wiki/Dwayne_Johnson"
         target="_blank">

        
      </a>
    </center>
  </body>
</html>
```

HTML Entity

HTML entity is a groupd of characters starts with '&' and ends with semicolon(;).
 HTML entities are used to display reversed characters or hidden characters in html.
 We have following list of html entities.

ex:

<u>HTML entity</u>	<u>character</u>
>	>
<	<
«	<<
»	>>
 	(1 space)
©	
and etc.	

Q) What is difference between HTML tag and HTML element?

HTML tag

HTML tag starts with '<' symbol and ends with '>' symbol.

ex: <html>,<head>,<body>,<h1> and etc.

HTML element

HTML element defines opening tag, some content and closing tag.

ex: <h1> This is heading tag </h1>
<p> This is paragraph tag </p>

Q) What is the different between block elements and inline elements?

block elements

A block elements always start with new line.

They will occupy 100% of width.

ex: <h1> , <p> , <div> and etc

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <h1> This is heading tag </h1>
        <p> This is paragraph tag </p>
        <div> This is division tag </div>
    </body>
</html>
```

inline elements

Inline elements starts with same line.

They will occupy width as much as required.

ex: ,<i>, and etc.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <b>This is bold tag</b>
        <i>This is italic tag</i>
        <span>This is span tag</span>
    </body>
</html>
```

Q) Types of tags in HTML ?

We have two types of tags in HTML.

1) Paired Tags / Container Tags

Paired tag contains opening tag and closing tag.

ex: <html>, <head>, <body> , <title> and etc.

2) Unpaired Tags / Empty Tags

Unpaired tag contains only opening and does not have any closing tag.

ex:
 , <hr>, , <link> and etc.

Q) Types of list in HTML ?

We have three types of list in HTML.

- 1) Ordered list
- 2) Unordered list
- 3) Description list

Ordered list

A tag is used to represent ordered list with numeric and alphabet.

Order list contains list of items.

Each list of item we can represent by using tag.

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    Courses:
    <ol>
      <li>ReactJS</li>
      <li>AngularJS</li>
      <li>VueJS</li>
      <li>ExpressJS</li>
      <li>NodeJS</li>
      <li>NextJS</li>
    </ol>
  </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    Courses:
    <ol start="101">
      <li>ReactJS</li>
      <li>AngularJS</li>
      <li>VueJS</li>
      <li>ExpressJS</li>
      <li>NodeJS</li>
      <li>NextJS</li>
    </ol>
  </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    Courses:
    <ol type="a">
      <li>ReactJS</li>
      <li>AngularJS</li>
      <li>VueJS</li>
      <li>ExpressJS</li>
      <li>NodeJS</li>
      <li>NextJS</li>
    </ol>
  </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    Courses:
    <ol type="A">
      <li>ReactJS</li>
      <li>AngularJS</li>
      <li>VueJS</li>
      <li>ExpressJS</li>
      <li>NodeJS</li>
      <li>NextJS</li>
    </ol>
  </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    Courses:
    <ol type="i">
      <li>ReactJS</li>
      <li>AngularJS</li>
      <li>VueJS</li>
```

```

<li>ExpressJS</li>
<li>NodeJS</li>
<li>NextJS</li>
</ol>
</body>
</html>

```

Unordered list

A `` tag is used to represent unordered list with bullets.

Unorder list contains list of items.

Each list of item we can represent by using `` tag.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        Courses:
        <ul type="disc">
            <li>ReactJS</li>
            <li>AngularJS</li>
            <li>VueJS</li>
            <li>ExpressJS</li>
            <li>NodeJS</li>
            <li>NextJS</li>
        </ul>
    </body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        Courses:
        <ul type="circle">
            <li>ReactJS</li>
            <li>AngularJS</li>
            <li>VueJS</li>
            <li>ExpressJS</li>
            <li>NodeJS</li>
            <li>NextJS</li>
        </ul>
    </body>
</html>

```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    Courses:
    <ul type="square">
      <li>ReactJS</li>
      <li>AngularJS</li>
      <li>VueJS</li>
      <li>ExpressJS</li>
      <li>NodeJS</li>
      <li>NextJS</li>
    </ul>
  </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    Courses:
    <ul type="none">
      <li>ReactJS</li>
      <li>AngularJS</li>
      <li>VueJS</li>
      <li>ExpressJS</li>
      <li>NodeJS</li>
      <li>NextJS</li>
    </ul>
  </body>
</html>
```

Description list

A `<dl>` tag is used to represent description list.

A description list contains description term and description definition.

A `<dt>` tag is used to represent description term.

A `<dd>` tag is used to represent description definition.

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
```

```

<dl>
    <dt>HTML5</dt>
    <dd>
        HTML5 is a advanced hypertext markup language
        which is used to develop light weight components
        to design interactive static web pages and web
        applications.
    </dd>
</dl>
</body>
</html>

```

HTML Forms

HTML forms are used to accept the values from the user.

It will send the data to database or server for processing.

To create a form in html we need to use <form> tag.

We have following list of form components.

ex: Label, TextField, Buttons, Checkbox, RadioButton, textarea, select box, and etc.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <form>
            <label>Name:</label>
            <input type="text" name="t1"/>
            <br>
            <label>Password:</label>
            <input type="password" name="t2"/>
            <br>
            <label>Age:</label>
            <input type="number" name="t3"/>
            <br>
            <label>DOB:</label>
            <input type="date" name="t4"/>
            <br>
            <label>Gender:</label>
            <input type="radio" value="male" name="t5"/>MALE
            <input type="radio" value="female" name="t5"/>FEMALE
            <br>
            <label>Marital Status:</label>
            <input type="checkbox" value="married" name="t6"/>MARRIED
            <input type="checkbox" value="single" name="t6"/>SINGLE
            <br>
            <label>Qualification:</label>
            <select name="t7">
                <option value="">none</option>

```

```

        <option value="b.tech">B.Tech</option>
        <option value="b.sc">B.SC</option>
        <option value="b.com">B.COM</option>
    </select>
    <br>
    <label> Address : <label>
    <textarea name="t8" rows="5" cols="10"></textarea>
    <br>
    <input type="reset" value="reset"/>
    <input type="submit" value="submit"/>
</form>
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <form>
            <label>Name:</label>
            <input type="text" name="t1" autocomplete="off" required/>
            <br>
            <label>Password:</label>
            <input type="password" name="t2" required/>
            <br>
            <label>Age:</label>
            <input type="number" name="t3" required/>
            <br>
            <label>DOB:</label>
            <input type="date" name="t4" required/>
            <br>
            <label>Gender:</label>
            <input type="radio" value="male" name="t5"/>MALE
            <input type="radio" value="female" name="t5"/>FEMALE
            <br>
            <label>Marital Status:</label>
            <input type="checkbox" value="married" name="t6"/>MARRIED
            <input type="checkbox" value="single" name="t6"/>SINGLE
            <br>
            <label>Qualification:</label>
            <select name="t7">
                <option value="">none</option>
                <option value="b.tech">B.Tech</option>
                <option value="b.sc">B.SC</option>
                <option value="b.com">B.COM</option>
            </select>
            <br>

```

```

<label> Address : <label>
<textarea name="t8" rows="5" cols="10"></textarea>
<br>
<input type="reset" value="reset"/>
<input type="submit" value="submit"/>
</form>
</body>
</html>

```

ex:1

index.html

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <form name="myform" action="a.html" method="POST">
            <label>Name:</label>
            <input type="text" name="t1" autocomplete="off" required/>
            <br>
            <label>Password:</label>
            <input type="password" name="t2" required/>
            <br>
            <label>Age:</label>
            <input type="number" name="t3" required/>
            <br>
            <label>DOB:</label>
            <input type="date" name="t4" required/>
            <br>
            <label>Gender:</label>
            <input type="radio" value="male" name="t5"/>MALE
            <input type="radio" value="female" name="t5"/>FEMALE
            <br>
            <label>Marital Status:</label>
            <input type="checkbox" value="married" name="t6"/>MARRIED
            <input type="checkbox" value="single" name="t6"/>SINGLE
            <br>
            <label>Qualification:</label>
            <select name="t7">
                <option value="">none</option>
                <option value="b.tech">B.Tech</option>
                <option value="b.sc">B.SC</option>
                <option value="b.com">B.COM</option>
            </select>
            <br>
            <label> Address : <label>
            <textarea name="t8" rows="5" cols="10"></textarea>
            <br>
            <input type="reset" value="reset"/>
        </form>
    </body>
</html>

```

```

        <input type="submit" value="submit"/>
    </form>
</body>
</html>

```

a.html

```

<!DOCTYPE html>
<html>
    <head>
        <title>A.html</title>
    </head>
    <body bgcolor="yellow">

        <center>welcome to A.html document </center>
    </body>
</html>

```

ex:2

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <form name="myform" action="#" method="GET">
            <input type="text" name="t1" autocomplete="off" placeholder="username"/>
<br>
            <input type="password" name="t2" placeholder="password"/> <br>
            <input type="submit" value="Login"/>
        </form>
    </body>
</html>

```

HTML Table

A table is used to store the data in rows and columns.

A `<table>` tag is used to represent table in html.

A `<tr>` tag is used represent table row.

A `<th>` tag is used represent table heading.

A `<td>` tag is used to represent table data.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <table border="0">
            <tr>
                <th>NO</th>
                <th>NAME</th>

```

```

        <th>ADDRESS</th>
    </tr>
    <tr>
        <td>101</td>
        <td>Alan</td>
        <td>Texas</td>
    </tr>
    <tr>
        <td>102</td>
        <td>Jose</td>
        <td>Chicago</td>
    </tr>
    <tr>
        <td>103</td>
        <td>Jason</td>
        <td>Florida</td>
    </tr>
</table>
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <table border="1" align="center" width="100%" bgcolor="cyan">
            <caption>Employee Details</caption>
            <tr>
                <th>NO</th>
                <th>NAME</th>
                <th>ADDRESS</th>
            </tr>
            <tr>
                <td>101</td>
                <td>Alan</td>
                <td>Texas</td>
            </tr>
            <tr>
                <td>102</td>
                <td>Jose</td>
                <td>Chicago</td>
            </tr>
            <tr>
                <td>103</td>
                <td>Jason</td>
                <td>Florida</td>
            </tr>
        </table>
    </body>
</html>

```

```
        </table>
    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <table border="1" align="center" cellspacing="10px" cellpadding="10px">
            <caption>Employee Details</caption>
            <tr bgcolor="cyan">
                <th>NO</th>
                <th>NAME</th>
                <th>ADDRESS</th>
            </tr>
            <tr>
                <td>101</td>
                <td>Alan</td>
                <td>Texas</td>
            </tr>
            <tr>
                <td>102</td>
                <td>Jose</td>
                <td>Chicago</td>
            </tr>
            <tr>
                <td>103</td>
                <td>Jason</td>
                <td>Florida</td>
            </tr>
        </table>
    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <table border="1" align="center">
            <caption>Employee Details</caption>
            <tr bgcolor="cyan">
                <th>NO</th>
                <th>NAME</th>
```

```

        <th>ADDRESS</th>
    </tr>
    <tr>
        <td>101</td>
        <td>Alan</td>
        <td>Texas</td>
    </tr>
    <tr>
        <td>102</td>
        <td>Jose</td>
        <td>Chicago</td>
    </tr>
    <tr>
        <td>103</td>
        <td>Jason</td>
        <td>Florida</td>
    </tr>
    <tr>
        <td colspan="3">Thank you</td>
    </tr>
</table>
</body>
</html>

```

HTML form using Table

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <form name="myform" action="#" method="POST">
            <table align="center">
                <tr>
                    <td><label>UserName:</label></td>
                    <td><input type="text" name="t1" autocomplete="off"
required/></td>
                </tr>
                <tr>
                    <td><label>Password:</label></td>
                    <td><input type="password" name="t2"
required/></td>
                </tr>
                <tr>
                    <td><label>Qualification:</label></td>
                    <td>
                        <select name="t3">
                            <option value="">none</option>
                            <option
value="btech">B.TECH</option>

```

```

        <option value="bsc">B.SC</option>
        <option
value="bcom">B.COM</option>
            </select>
        </td>
    </tr>
    <tr>
        <td><input type="reset" value="reset"/></td>
        <td><input type="submit" value="submit"/></td>
    </tr>
</table>
</form>
</body>
</html>

```

Datalist tag

A `<datalist>` tag/element is used to specify list of predefine options for an `<input>` tag/element.

A `<datalist>` tag/element provides autocomplete features for an `<input>` element/tag.

User will see a drop-down list of predefine options for an `<input>` tag/element.

A `<datalist>` tag/element "id" attribute must be same as `<input>` tag/element "list" attribute.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        Select Country : <input type="text" list="countries"/>

        <datalist id="countries">
            <option value="India">
            <option value="Ireland">
            <option value="Iran">
            <option value="Iraq">
            <option value="Iceland">
        </datalist>
    </body>
</html>

```

<details> and <summary> tag

A `<details>` tag/element is used display special content where a user can open and close on demand.

A `<details>` tag is used to design interactive widgets where user can open and close.

A `<details>` tag contains `<summary>` tag.

We can keep any sort of tags inside `<details>` tags.

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <details>
      <summary>HTML</summary>
      <h1>HTML is a markup language</h1>
      <p>It is a case insensitive language</p>
      <div>It is a tag based language</div>
    </details>
  </body>
</html>
```

<big> tag

The `<big>` tag is used to make the text one size bigger i.e from small to medium, medium to large, large to x-large.

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <p>This is paragraph tag</p>
    <p>This is <big>paragraph</big> tag</p>
  </body>
</html>
```

<small> tag

The `<small>` HTML element represents small print like copyright and legal text.

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <small>&copy;AllRight Reserved-2023</small>
  </body>
</html>
```

<cite> tag

The `<cite>` tag defines the title of a creative work i.e movie , songs, Poems and etc.

The text in the `<cite>` element usually renders in italic.

ex:

```
<!DOCTYPE html>
```

```
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    Movie Title : <cite>Salaar</cite>
  </body>
</html>
```

<mark>

The HTML `<mark>` tag is used to mark or highlight text that has special interest.

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <h1>This is is <mark>very important</mark> for practice</h1>
  </body>
</html>
```

<sub> tag

The `<sub>` tag defines subscript text.

Subscript text appears half a character below the normal line, and is sometimes rendered in a smaller font.

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <h1>H<sub>2</sub>O</h1>
  </body>
</html>
```

<sup> tag

The `<sup>` tag defines superscript text.

Superscript text appears half a character above the normal line, and is sometimes rendered in a smaller font.

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <h1>x<sup>2</sup>+y<sup>2</sup></h1>
  </body>
```

```
</html>
```

<bdo> tag

BDO stands for Bi-Directional Override.

The <bdo> tag is used to override the current text direction.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <bdo dir="rtl"> Hello </bdo>
    </body>
</html>
```

HTML <header> and <nav> tag

The <header> element represents a container for introductory content or a set of navigational links.

The <nav> tag defines a set of navigation links.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <header style="background-color:cyan">
            <center>
                <nav>
                    <a href="" style="text-decoration:none"> HOME
                    <a href="" style="text-decoration:none"> ABOUTUS
                    <a href="" style="text-decoration:none"> SERVICES
                    <a href="" style="text-decoration:none"> PORTFOLIO
                    <a href="" style="text-decoration:none"> CONTACT
                </nav>
            </center>
        </header>
    </body>
</html>
```

HTML <section>, <article> and <figure> tag

The <section> tag defines a section in a document.

The <article> tag specifies independent, self-contained content.

The <figure> tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <section>
      <article>
        <center>
          <h1>Hollywood Actor</h1>

          <figure>
            
            <figcaption>Dwayne Johnson</figcaption>
          </figure>
          <p>
            Dwayne Douglas Johnson (born May 2, 1972), also
            known by his ring name the Rock, is an American actor, film producer, and retired
            professional wrestler. Widely regarded as one of the greatest professional wrestlers of all
            time,[6][7] he was integral to the development and success of the World Wrestling Federation
            (WWF, now WWE) during the Attitude Era, an industry boom period in the late 1990s and
            early 2000s. Johnson wrestled for the WWF for eight years before pursuing an acting career.
            His films have grossed over $3.5 billion in North America and over $10.5 billion
            worldwide,[8] making him one of the world's highest-grossing and highest-paid actors.
          </p>
        </center>
      </article>
    </section>
  </body>
</html>
```

HTML <footer> tag

The <footer> tag defines a footer for a document or section.

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <footer>
      <center>
        <small>&copy;All Right reserved - 2023</small>
      </center>
    </footer>
  </body>
</html>
```

NotePad++ Editor [Download link](https://notepad-plus-plus.org/downloads/) : <https://notepad-plus-plus.org/downloads/>

HTML Frames

To use frames on a web page we need to use <frameset> tag instead of <body> tag.
The <frameset> tag defines how to divide the window into frames.
The "rows" attribute of <frameset> tag defines horizontal frames.
The "cols" attribute of <frameset> tag defines vertical frames.
Each frame is indicated by <frame> tag and it defines which document should be open into that place.

ex:1

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <frameset cols="33%,*,33%">
        <frame src="a.html"/>
        <frame src="b.html"/>
        <frame src="c.html"/>
    </frameset>
</html>
```

a.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>A.html</title>
    </head>
    <body bgcolor="red">
        </body>
</html>
```

b.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>B.html</title>
    </head>
    <body bgcolor="green">
        </body>
</html>
```

c.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>C.html</title>
    </head>
```

```
<body bgcolor="blue"></body>
</html>
```

ex:2

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <frameset rows="33%,*,33%">
        <frame src="a.html"/>
        <frame src="b.html"/>
        <frame src="c.html"/>
    </frameset>
</html>
```

a.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>A.html</title>
    </head>
    <body bgcolor="orange"></body>
</html>
```

b.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>B.html</title>
    </head>
    <body bgcolor="white">
        <center>
            
        </center>
    </body>
</html>
```

c.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>C.html</title>
    </head>
    <body bgcolor="green">
        </body>
</html>
```

HTML <iframe> tag

It is used to specify inline frame.

A <iframe> tag/element is used to embed a document into current HTML document.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <iframe src="http://www.ihubtalent.com" width="400px"
height="400px"/>
    </body>
</html>
```

Steps to display Google Map on a web page

step1: Goto Google Maps.

ex: https://www.google.com/maps

step2: Type Ihub Talent in Google Map Search Box.

step3: Click on "menu" button.

step4: Click on "share and embed Map"

step5: Click on "embeded a map" link.

step6: Click to "Copy Html".

step7: Paste the code inside <body> tag of index.html file.

step8: Check the output on browser window.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <iframe
src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d3806.487656502131!2
d78.44202777415968!3d17.436358401394536!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3
m3!1m2!1s0x3bcb919633f78bef%3A0xbb63515db9bc2eba!2siHub%20Talent%20Info%20S
ystems%20-
%20Testing%20Tools%2C%20Full%20Stack%20Java%2C%20Python%20Training%20Insti
tute%20in%20Hyderabad!5e0!3m2!1sen!2sin!4v1694234739077!5m2!1sen!2sin"
width="100%" height="450" style="border:0;" allowfullscreen=""
loading="lazy" referrerpolicy="no-referrer-when-downgrade"></iframe>
    </body>
</html>
```

HTML <audio> tag

The HTML <audio> tag/element is used to play an audio file on a web page.

<audio> tag contains "controls" attribute adds audio control like play,pause ,volume and etc.

The <source > tag/element allows us to specify alternate audio file which the browser my choose.

HTML audio formats can be MP3,WAV,OGG and etc.

HTML Audio Media types are

<u>File Formats</u>	<u>Media Type</u>
MP3	audio/mpeg
OGG	audio/ogg
WAV	audio/wav

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB</title>
    </head>
    <body>
        <audio controls>
            <source src="media/sajda.mp3" type="audio/mpeg">
        </audio>
    </body>
</html>
```

HTML <video> tag

The <video> tag or element is used to embed video content in a document such as movie clip, other video streams.

The <video> tag contains one or more <source> tag with different video source.

There are three supported video fromats in html are MP4,webM and OGG.

HTML video media types are

<u>File format</u>	<u>Media Type</u>
MP4	video/mp4
OGG	video/ogg

and etc.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB</title>
    </head>
    <body>
        <video controls width="200px" height="200px">
            <source src="media/video.mp4" type="video/mp4">
        </video>
    </body>
</html>
```

Graphics in HTML5

There are two ways to draw the graphics in HTML5.

- 1)SVG
- 2)CANVAS

1) SVG

HTML5 uses SVG technology to derived graphics in HTML.

SVG stands for Scalable Vector Graphics.

SVG is used to draw two-dimensional vector based graphics in HTML.

World Wide Web Consortium (W3C) prefers SVG technology to draw the graphics in HTML.

A <svg> tag/element is a container tag for vector graphics.

A <svg> tag/element contains various methods to display circle, rectangle, polygon, graphic images and etc.

ex:1

```
<!DOCTYPE html>
<html>
    <head>
        <title>INDEX</title>
    </head>
    <body>
        <!-- container -->
        <svg style="border:2px solid black" width="300px" height="300px">
            </svg>
    </body>
</html>
```

ex:2

```
<!DOCTYPE html>
<html>
    <head>
        <title>INDEX</title>
    </head>
    <body>
        <!-- container -->
        <svg style="border:2px solid black" width="300px" height="300px">
            <!-- circle -->
            <circle cx="150px" cy="150px" r="50px"/>
        </svg>
    </body>
</html>
```

ex:3

```
<!DOCTYPE html>
<html>
    <head>
        <title>INDEX</title>
    </head>
    <body>
        <!-- container -->
        <svg style="border:2px solid black" width="300px" height="300px">
            <!-- circle -->
            <circle cx="150px" cy="150px" r="50px"
                fill="#FFFF00" stroke="#FF0000" stroke-width="5px"/>
        </svg>
    </body>
</html>
```

```

        </body>
    </html>

ex:4
<!DOCTYPE html>
<html>
    <head>
        <title>INDEX</title>
    </head>
    <body>
        <!-- container -->
        <svg style="border:2px solid black" width="300px" height="300px">
            <!-- reactangle -->
            <rect x="50px" y="20px" width="200px" height="200px"
                fill="green" stroke="blue" stroke-width="5px"/>
        </svg>
    </body>
</html>
```

```

ex:5
<!DOCTYPE html>
<html>
    <head>
        <title>INDEX</title>
    </head>
    <body>
        <!-- container -->
        <svg style="border:2px solid black" width="300px" height="300px">
            <!-- polygon -->
            <polygon points="10,78 100,10 250,190 160,210"
                fill="yellow" stroke="red" stroke-width="5px"/>
        </svg>
    </body>
</html>
```

2) CANVAS

A <canvas> tag is used to draw vector graphics via javascript.

A <canvas> tag/element is a container tag.

A <canvas> tag/element contains various methods to display lines, circle, rectangle, polygon, graphic images and etc.

ex:1

```

<!DOCTYPE html>
<html>
    <head>
        <title>INDEX</title>
    </head>
    <body>
        <!-- container -->
        <canvas style="border:2px solid black;" width="300px" height="300px">
```

```

        </canvas>
    </body>
</html>

ex:2
<!DOCTYPE html>
<html>
    <head>
        <title>INDEX</title>
    </head>
    <body>
        <!-- container -->
        <canvas id="myId" style="border:2px solid black;" width="300px" height="300px">
            </canvas>
        <script>
            var context=document.getElementById("myId");
            var c=context.getContext("2d");
            c.moveTo(0,0);
            c.lineTo(400,400);
            c.stroke();
        </script>
    </body>
</html>

ex:2
<!DOCTYPE html>
<html>
    <head>
        <title>INDEX</title>
    </head>
    <body>
        <!-- container -->
        <canvas id="myId" style="border:2px solid black;" width="300px" height="300px">
            </canvas>
        <script>
            var context=document.getElementById("myId");
            var c=context.getContext("2d");
            c.beginPath();
            c.arc(100,100,50,0,2*Math.PI);
            c.stroke();
        </script>
    </body>
</html>
```

```

ex:3
<!DOCTYPE html>
<html>
    <head>
        <title>INDEX</title>
    </head>
    <body>
        <!-- container -->
        <canvas id="myId" style="border:2px solid black;" width="300px" height="300px">
        </canvas>

        <script>
            var context=document.getElementById("myId");
            var c=context.getContext("2d");
            c.font = "30px Arial";
            c.fillText("Hello World", 20, 50);
        </script>
    </body>
</html>

```

Q) What is the difference between HTML and HTML5 ?

HTML

To represent html document we need to Use <!DOCTYPE>.

HTML is bit slow.

It is inflexible for the programmer.

It does not support drag and drop menu.

It is less mobile friendly.

It does not allow to execute javascript.

We can't play audio and video without using flash player.

Vector graphics is possible by using technologies VML, silver light , adope flash and etc.

HTML5

To represent html5 document we need to use <!DOCTYPE html>.

HTML5 is more faster because it is light weight.

It is flexible for the programmer.

It supports drag and drop menu.

It is more mobile friendly.

It allows to execute javascript with help of JS Web work API.

We can play audio and video with the help <audio> and <video> tag without using flash player.

Vector graphics is possible by using internal technologies like SVG and CANVAS.

It can't handle inaccurate errors.

It can handle inaccurate errors.

Shapes like circle, rectangle , polygon and etc are not easy to draw.

Shapes like circle, rectangle, polygon and etc are easy to draw.

Q) What is the difference between <div> tag and tag ?

div

It is a block element.

It is used to wrap the sections.

It is used to develop css based layouts.

span

It is a inline element.

It is used to wrap the small portion of a text, images and etc.

It is used to stylize the text.

Q) What is the disadvantages of HTML5?

1. It is used to develop static web pages but not dynamic web pages.
2. Security features are not good.
3. For simple design we need to do lot of code.
4. If code is increases then it will increase the complexity.

How do HTML document will execute in a browser window

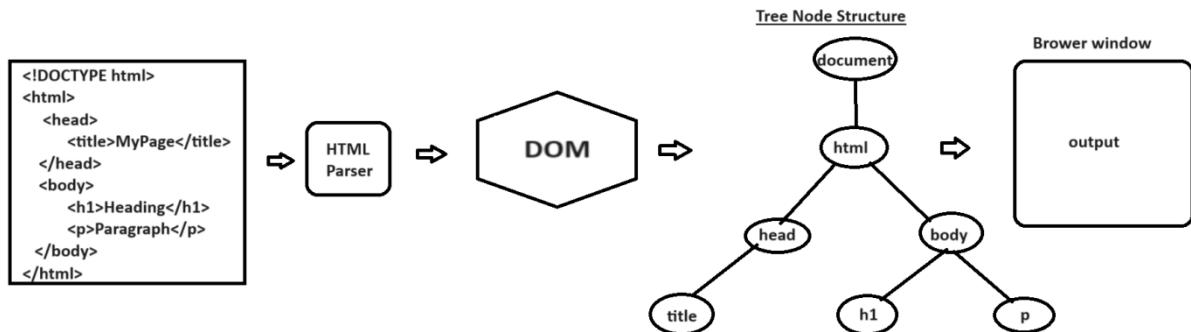


Diagram: class7.1

HTML parse will parse the data from HTML document and it will give to DOM.

DOM stands for Document Object Mode.

DOM is present in a browser window.

DOM will convert our HTML document to Tree Node structure.

Late, Tree Node structure will execute in a browser window.

CSS

CSS stands for Cascading Styles Sheet.

CSS is used to apply the styles on elements.

The latest version of CSS3 was introduced in 2001.

The main objective of CSS3 are.

1. To set the positioning of an element.
2. To apply the styles to describe how an element should look like.
3. To perform some sort of animations.

syntax:

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            -
            -
            -
        </style>
    </head>
    <body>
        </body>
</html>
```

Note: Here style tag will cascade from head to body.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                color:#0000ff;
                background-color:#FFFF00;
            }
        </style>
    </head>
    <body>
        <h1>This is Heading Tag</h1>
    </body>
</html>
```

ex:2

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            body
```

```

        {
            background-color:#00FF00;
        }
    h1
    {
        color:#0000ff;
        font-size:50px;
    }

    </style>
</head>
<body>
    <h1>This is Heading Tag</h1>
</body>
</html>

```

CSS syntax

CSS syntax contains selector and declaration block.

ex:

```

selector
|
h1{color:#0000ff;font-size:50px;}
|-----|
|
declaration block

```

CSS declaration block may contains multiple CSS properties.

Each css property contains property name and property value seperated with colon.

Each property ends with semicolon.

Advantages of CSS

- 1) It is easy to learn and easy to use.
- 2) It saves lot of development time.
- 3) It supported by all major browsers.
- 4) It supports global change.
- 5) Flexibility

Disadvantages of CSS

- 1) Fragmentation
- 2) Need to update all the versions of CSS.

Types of CSS

We have three types of CSS.

- 1)Inline CSS
- 2)Internal CSS / Embedded CSS
- 3)External CSS / Seperate CSS

1. Inline CSS

Inline CSS is used to apply unique styles on single element.

Using "style" attribute we can achieve inline CSS.

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <h1 style="color:red;">This is Heading Tag1</h1>
    <h1 style="background-color:yellow;">This is Heading Tag 2</h1>
  </body>
</html>
```

2. Internal CSS

It is used to apply unique styles on single web page.
Using `<style>` tag we can achieve internal CSS.

A `<style>` tag we need to declare inside `<head>` tag.

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style type="text/css">
      h1
      {
        color:blue;
        background-color:yellow;
      }
      p
      {
        font-size:50px;
        text-align:center;
      }
    </style>
  </head>
  <body>
    <h1>This is Heading Tag1</h1>
    <h1>This is Heading Tag2</h1>
    <p>This is paragraph tag</p>
  </body>
</html>
```

3. External CSS

It is used to apply the unique styles on entire web site (collection of web pages).

In external css, we need to create two files

- i) .html file
- ii) .css file

In external css , we will maintain html code in .html file and CSS code in .css file.

It is not possible to execute .css file seperately in a browser window.

In order to execute .css file we need to link css file with HTML file using `<link>` tag.

ex:

```
index.html
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <!-- add external css -->
        <link rel="stylesheet" href="mystyles.css">
    </head>
    <body>
        <h1> This is External CSS </h1>
    </body>
</html>
```

```
mystyles.css
body
{
    background-color:cyan;
}
h1
{
    color:blue;
    text-align:center;
}
```

CSS background property

CSS background property is used to set the background in a web page.
If we set the background to body then it will reflect to entire web page.
If we want we can apply background to elements also.

ex: <h1>, <p>, <div> and etc.

We have following list of CSS background properties.

- 1) background-color
- 2) background-image
- 3) background-repeat
- 4) background-size
- 5) background-position
- 6) background-attachment
- 7) background shorthand
- 8) background-blend-mode

and etc.

1) background-color

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            body
            {
                background-color:yellow;
            }
        </style>
    </head>
    <body>
```

```

        </style>
    </head>
    <body>
    </body>
</html>

ex:
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                background-color:yellow;
            }
        </style>
    </head>
    <body>
        <h1> This is Heading Tag</h1>
    </body>
</html>
```

2) background-image

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                background-image: url("images/bg.jpg");
            }
        </style>
    </head>
    <body>
        <h1> This is Heading Tag</h1>
    </body>
</html>
```

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            body
            {
                background-image: url("images/bg.jpg");
            }
        </style>
    </head>
    <body>
```

```

        }
    </style>
</head>
<body>
    <h1> This is Heading Tag</h1>
</body>
</html>

```

3) background-repeat

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            body
            {
                background-color:#FFFFFF;
                background-image: url("images/bg.jpg");
                background-repeat:no-repeat;
            }
        </style>
    </head>
    <body>
        <h1> This is Heading Tag</h1>
    </body>
</html>

```

4) background-size

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            body
            {
                background-color:#FFFFFF;
                background-image: url("images/bg.jpg");
                background-repeat:no-repeat;
                background-size: 1500px 1100px;
            }
        </style>
    </head>
    <body>
        <h1> This is Heading Tag</h1>
    </body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>

```

```

<head>
    <title>MyPage!</title>
    <style>
        body
        {
            background-color:#FFFFFF;
            background-image: url("images/bg.jpg");
            background-repeat:no-repeat;
            background-size: cover;
        }
    </style>
</head>
<body>
    <h1> This is Heading Tag</h1>
</body>
</html>

```

5) background-position

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            body
            {
                background-color:#FFFFFF;
                background-image: url("images/bg.jpg");
                background-repeat:no-repeat;
                background-size: static;
                background-position: center 0px;
            }
        </style>
    </head>
    <body>
        <h1> This is Heading Tag</h1>
    </body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            body
            {
                background-color:#FFFFFF;
                background-image: url("images/bg.jpg");
                background-repeat:no-repeat;
                background-size: static;

```

```
        background-position: right 0px;
    }
</style>
</head>
<body>
    <h1> This is Heading Tag</h1>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            body
            {
                background-color:#FFFFFF;
                background-image: url("images/bg.jpg");
                background-repeat:no-repeat;
                background-size: static;
                background-position: left 0px;
            }
        </style>
    </head>
    <body>
        <h1> This is Heading Tag</h1>
    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            body
            {
                background-color:#FFFFFF;
                background-image: url("images/bg.jpg");
                background-repeat:no-repeat;
                background-size: static;
                background-position: center 200px;
            }
        </style>
    </head>
    <body>
        <h1> This is Heading Tag</h1>
    </body>
</html>
```

6) background-attachment

ex:

—

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
```

7) background shorthand

```
    background-color:#FFFFFF;  
    background-image: url("images/bg.jpg");  
    background-repeat:no-repeat;  
    background-position: center 200px;  
    background-attachment:fixed;
```

or

background: #FFFFFF url("images/bg.jpg") no-repeat center 200px fixed;

ex:

8) background-blend-mode

```
<!DOCTYPE html>
```

<html>

<head>

<title>MyPage!</title>

<style>

```
body  
{
```

background-color:#FFFFFF, #FFFFFF;

```

background-image: url("images/bg.jpg"),
url("images/micky.png");
background-repeat:no-repeat, no-repeat ;
background-size: cover , 400px;
background-position: center 0px , center 100px;
background-attachment:fixed , fixed;
background-blend-mode: lighten;
}

</style>

</head>
<body>

</body>
</html>

ex:
<!DOCTYPE html>
<html>
<head>
<title>MyPage!</title>

<style>
body
{
background-color:#FFFFFF, #FFFFFF;
background-image: url("images/bg.jpg"),
url("images/micky.png");
background-repeat:no-repeat, no-repeat ;
background-size: cover , 400px;
background-position: center 0px , center 100px;
background-attachment:fixed , fixed;
background-blend-mode: darken;
}

</style>

</head>
<body>

</body>
</html>

```

CSS border property

CSS border property is used to add the border to elements.

We have following list of CSS border properties.

- 1) border-style
- 2) border-color
- 3) border-width
- 4) border shorthand

1) border-style

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                border-style: solid;
            }
            h2
            {
                border-style: dotted;
            }
            h3
            {
                border-style: dashed;
            }
            p
            {
                border-style:double;
            }
            div
            {
                border-style:inset;
            }
        </style>
    </head>
    <body>
        <h1> Heading Tag 1 </h1>
        <h2> Heading Tag 2 </h2>
        <h3> Heading Tag 3 </h3>
        <p> This is paragraph </p>
        <div> This is Division tag </div>
    </body>
</html>
```

2) border-color

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                border-style: solid;
            }
        </style>
    </head>
    <body>
        <h1> Heading Tag 1 </h1>
        <h2> Heading Tag 2 </h2>
        <h3> Heading Tag 3 </h3>
        <p> This is paragraph </p>
        <div> This is Division tag </div>
    </body>
</html>
```

```

        border-color: red;
    }
h2
{
    border-style: dotted;
    border-color:blue;
}
h3
{
    border-style: dashed;
    border-color:green;
}
p
{
    border-style:double;
    border-color:yellow;
}
div
{
    border-style:inset;
    border-color:pink;
}

```

</style>

</head>

<body>

<h1> Heading Tag 1 </h1>

<h2> Heading Tag 2 </h2>

<h3> Heading Tag 3 </h3>

<p> This is paragraph </p>

<div> This is Division tag </div>

</body>

</html>

3) border-width

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                border-style: solid;
                border-color: red;
                border-width: 5px;
            }
            h2

```

```

        {
            border-style: dotted;
            border-color:blue;
            border-width: 10px;
        }
    h3
    {
        border-style: dashed;
        border-color:green;
        border-width: 15px;
    }
    p
    {
        border-style:double;
        border-color:yellow;
        border-width: 20px;
    }
    div
    {
        border-style:inset;
        border-color:pink;
        border-width: 25px;
    }

```

}

</style>

</head>

<body>

<h1> Heading Tag 1 </h1>

<h2> Heading Tag 2 </h2>

<h3> Heading Tag 3 </h3>

<p> This is paragraph </p>

<div> This is Division tag </div>

</body>

</html>

4) border shorthand

border-width : 2px;
 border-style : solid;
 border-color : blue;

or

border : 2px solid blue;

ex:

<!DOCTYPE html>

```

<html>
  <head>
    <title>MyPage!</title>
    <style>
      h1
      {
        border : 2px solid red;
      }
      h2
      {
        border : 2px dotted blue;
      }
      h3
      {
        border : 2px dashed green;
      }
      p
      {
        border : 2px double yellow;
      }
      div
      {
        border : 2px inset pink;
      }
    </style>
  </head>
  <body>
    <h1> Heading Tag 1 </h1>
    <h2> Heading Tag 2 </h2>
    <h3> Heading Tag 3 </h3>
    <p> This is paragraph </p>
    <div> This is Division tag </div>
  </body>
</html>

```

CSS Colors

In CSS , we can declare colors in following ways.

- 1) valid color name
- 2) Hexa value
- 3) RGB value
- 4) HSL value

1) valid color name

```

<!DOCTYPE html>
<html>
  <head>
```

```

<title>MyPage!</title>
<style>
    body
    {
        background-color: violet;
    }
    h1
    {
        color:forestgreen;
    }
</style>
</head>
<body>
    <h1> This is Heading Tag </h1>
</body>
</html>

```

2) Hexa value

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            body
            {
                background-color: #FFFF00;
            }
            h1
            {
                color:#00ff00;
            }
        </style>
    </head>
    <body>
        <h1> This is Heading Tag </h1>
    </body>
</html>

```

3) RGB value

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            body
            {
                background-color: rgb(255,0,0);
            }
            h1
            {

```

```

        color:rgb(0,0,255);
    }
</style>
</head>
<body>
    <h1> This is Heading Tag </h1>
</body>
</html>

```

4) HSL value

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            body
            {
                background-color: hsl(330, 100%, 60%);
            }
            h1
            {
                color: hsl(0, 100%, 40%);
            }
        </style>
    </head>
    <body>
        <h1> This is Heading Tag </h1>
    </body>
</html>

```

CSS margin property

CSS margin property is used to give the space around the elements, outside the defined border.

CSS margin contains following properties.

- 1) margin-top
- 2) margin-right
- 3) margin-bottom
- 4) margin-left
- 5) margin shorthand

All the above margin properties will take following values.

- i) auto : It declares the margin by the browser.
- ii) value : It declares the margin in px,pt,cm,em and etc.
- iii) % : It declares the margin in percentage.
- iv) inherit : It takes the margin from parent tag.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>

```

```

<style>
    h1
    {
        border:2px solid #000;
        margin-top:100px;
        margin-right:100px;
        margin-bottom:100px;
        margin-left:100px;
    }
</style>
</head>
<body>
    <h1> This is Heading Tag </h1>
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                border:2px solid #000;

                margin-top:10%;
                margin-left:20%;
                margin-right:20%;
                margin-bottom:10%;

            }
        </style>
    </head>
    <body>
        <h1> This is Heading Tag </h1>
    </body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                border:2px solid #FF0000;
            }
        </style>
    </head>
    <body>

```

```

        margin:50px;
    }
h1
{
    border:2px solid #000;
    margin:inherit;
}

```

</style>

</head>

<body>

<div>

<h1> This is Heading Tag </h1>

</div>

</body>

</html>

If margin contains four values.

ex:

margin : 25px 50px 100px 150px;

top , right, bottom , left

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                border:2px solid #000;
                margin: 25px 50px 100px 150px;
            }

```

</style>

</head>

<body>

<h1> This is Heading Tag </h1>

</body>

</html>

If margin contains three values.

ex:

margin : 50px 100px 150px;

top, right and left, bottom

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                border:2px solid #000;
                margin : 50px 100px 150px;
            }
        </style>
    </head>
    <body>
        <h1> This is Heading Tag </h1>
    </body>
</html>

```

If margin contains two values.

ex:

```

margin: 50px 100px;
top and bottom , left and right

```

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                border:2px solid #000;
                margin : 50px 100px;
            }
        </style>
    </head>
    <body>
        <h1> This is Heading Tag </h1>
    </body>
</html>

```

If margin contains one property:

ex:

```

margin : 100px;

```

All sides are 100px

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                border:2px solid #000;
                margin :100px;
            }

        </style>
    </head>
    <body>
        <h1> This is Heading Tag </h1>
    </body>
</html>

```

CSS border-radius property

The border-radius property defines the radius of the element's corners.
We have following list of border-radius properties.

- 1) border-top-left-radius
- 2) border-top-right-radius
- 3) border-bottom-right-radius
- 4) border-bottom-left-radius
- 5) border-radius shorthand

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                border:2px solid black;
                border-top-left-radius: 5px;
                border-top-right-radius:10px;
                border-bottom-left-radius:15px;
                border-bottom-right-radius:20px;
            }

        </style>
    </head>
    <body>
        <h1> This is Heading Tag </h1>
    </body>

```

```

</html>

ex:
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                border:2px solid black;
                border-radius : 5px 10px 15px 20px;
            }
        </style>
    </head>
    <body>
        <h1> This is Heading Tag </h1>
    </body>
</html>

```

Here border-radius goes to top left, top right, bottom right and bottom left.

```

ex:
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                border:2px solid black;
                border-radius : 5px 20px;
            }
        </style>
    </head>
    <body>
        <h1> This is Heading Tag </h1>
    </body>
</html>

```

Here border-radius goes to top left, bottom right, top right and bottom left.

```

ex:
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>

```

```

        h1
        {
            border:2px solid black;
            border-radius : 20px;
        }

    </style>
</head>
<body>
    <h1> This is Heading Tag </h1>

</body>
</html>

```

Here border-radius goes to equals for four sides.

CSS padding property

CSS padding property provide the space around the element content. inside the defined border.

We have following list of margin properties.

- 1) padding-top
- 2) padding-right
- 3) padding-bottom
- 4) padding-left
- 5) padding shorthand property

All the above properties will take following values.

- 1) auto : It describes the padding by the browser.
- 2) value : It describes the padding in px,pt,cm,em and etc.
- 3) % : It describes the padding in percentage.
- 4) inherit : It will take the padding from parent tag.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                border:2px solid red;
                padding-top:100px;
                padding-right:100px;
                padding-bottom:100px;
                padding-left:100px;
            }
        </style>
    </head>
    <body>
        <h1> Heading Tag</h1>
    </body>
</html>

```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      h1
      {
        border:2px solid red;
        padding-top:10%;
        padding-right:20%;
        padding-bottom:30%;
        padding-left:40%;
      }
    </style>
  </head>
  <body>
    <h1> Heading Tag</h1>
  </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      div
      {
        border:2px solid blue;
        padding-top:100px;
      }
      h1
      {
        border:2px solid red;
        padding-top:inherit;
      }
    </style>
  </head>
  <body>
    <div>
      <h1> Heading Tag</h1>
    </div>
  </body>
</html>
```

If padding contains four values.

ex: padding: 25px 50px 75px 100px;
 top , right, bottom , left

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      h1
      {
        border:2px solid red;
        padding: 25px 50px 75px 100px;
      }
    </style>
  </head>
  <body>
    <h1> Heading Tag</h1>
  </body>
</html>
```

If padding contains three values.

ex: padding: 25px 75px 100px;
top , left and right, bottom

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      h1
      {
        border:2px solid red;
        padding: 25px 75px 100px;
      }
    </style>
  </head>
  <body>
    <h1> Heading Tag</h1>
  </body>
</html>
```

If padding contains two values.

ex: padding: 75px 100px;
top and bottom , left and right

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
```

```

        h1
        {
            border:2px solid red;
            padding: 75px 100px;
        }
    </style>
</head>
<body>
    <h1> Heading Tag</h1>
</body>
</html>

```

If padding contains one value.

ex: padding: 50px;
 all sides are 50px.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                border:2px solid red;
                padding: 50px;
            }
        </style>
    </head>
    <body>
        <h1> Heading Tag</h1>
    </body>
</html>

```

CSS Text property

color

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                color:blue;
            }
        </style>
    </head>
    <body>
        <h1> Heading Tag</h1>
    </body>

```

background-color

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      h1
      {
        color:blue;
        background-color:yellow;
      }
    </style>
  </head>
  <body>
    <h1> Heading Tag</h1>
  </body>
</html>
```

text-align

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      h1
      {
        color:blue;
        background-color:yellow;
        text-align:right;
      }
    </style>
  </head>
  <body>
    <h1> Heading Tag</h1>
  </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      h1
      {
        color:blue;
        background-color:yellow;
      }
    </style>
  </head>
  <body>
    <h1> Heading Tag</h1>
  </body>
</html>
```

```
        text-align:center;
    }
</style>
</head>
<body>
    <h1> Heading Tag</h1>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>

            h1
            {
                color:blue;
                background-color:yellow;
                text-align:left;
            }
        </style>
    </head>
    <body>
        <h1> Heading Tag</h1>
    </body>
</html>
```

text-transform

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                color:blue;
                background-color:yellow;
                text-align:center;
                text-transform:uppercase;
            }
        </style>
    </head>
    <body>
        <h1> heading tag</h1>
    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      h1
      {
        color:blue;
        background-color:yellow;
        text-align:center;
        text-transform:lowercase;
      }
    </style>
  </head>
  <body>
    <h1> heading tag</h1>
  </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      h1
      {
        color:blue;
        background-color:yellow;
        text-align:center;
        text-transform:capitalize;
      }
    </style>
  </head>
  <body>
    <h1> heading tag</h1>
  </body>
</html>
```

text-decoration

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      h1
      {
        color:blue;
      }
    </style>
  </head>
  <body>
    <h1> heading tag</h1>
  </body>
</html>
```

```
background-color:yellow;
text-align:center;
text-transform:capitalize;
text-decoration:underline;
}
</style>
</head>
<body>
    <h1> heading tag</h1>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                color:blue;
                background-color:yellow;
                text-align:center;
                text-transform:capitalize;
                text-decoration:none;
            }
        </style>
    </head>
    <body>
        <h1> heading tag</h1>
    </body>
</html>
```

letter-spacing

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                color:blue;
                background-color:yellow;
                text-align:center;
                text-transform:capitalize;
                text-decoration:none;
                letter-spacing:4px;
            }
        </style>
    </head>
```

```
<body>
    <h1> heading tag</h1>
</body>
</html>

font-size
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                color:blue;
                background-color:yellow;
                text-align:center;
                text-transform:capitalize;
                text-decoration:none;
                letter-spacing:4px;
                font-size:80px;
            }
        </style>
    </head>
    <body>
        <h1> heading tag</h1>
    </body>
</html>
```

font-family

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                color:blue;
                background-color:yellow;
                text-align:center;
                text-transform:capitalize;
                text-decoration:none;
                letter-spacing:4px;
                font-size:80px;
                font-family:cursive;
            }
        </style>
    </head>
    <body>
```

```
<h1> heading tag</h1>
</body>
</html>

ex:
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                color:blue;
                background-color:yellow;
                text-align:center;
                text-transform:capitalize;
                text-decoration:none;
                letter-spacing:4px;
                font-size:80px;
                font-family:monospace;
            }
        </style>
    </head>
    <body>
        <h1> heading tag</h1>
    </body>
</html>
```

```
ex:
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                color:blue;
                background-color:yellow;
                text-align:center;
                text-transform:capitalize;
                text-decoration:none;
                letter-spacing:4px;
                font-size:80px;
                font-family:Fantasy ;
            }
        </style>
    </head>
    <body>
        <h1> heading tag</h1>
    </body>
```

```
</html>
```

font-weight

ex:

--

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>

            span
            {
                font-weight:bold;
            }

        </style>
    </head>
    <body>
        <span>This is span tag</span>
    </body>
</html>
```

ex:

--

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>

            span
            {
                font-weight:600;
            }

        </style>
    </head>
    <body>
        <span>This is span tag</span>
    </body>
</html>
```

CSS overflow property

The overflow property specifies what should happen if content overflow.

This property specifies whether to clip content or to add scrollbars when an element's content is too big to fit in a specified area.

Note: The overflow property works for block elements with a specified height.

value	Description
visible	The overflow is not clipped. It is rendered outside the element's box and

	it is default value.
hidden	the overflow is clipped and rest of the content will be invisible.
scroll	The overflow is clipped, but a scroll-bar is added to see the rest of the content.
auto	The overflow is clipped, a scroll-bar should be added to the rest of the content.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                width:200px;
                height:200px;
                border:2px solid black;
                overflow:visible;
            }
        </style>
    </head>
    <body>
        <div>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS). It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
        </div>
    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                width:200px;
                height:200px;
                border:2px solid black;
                overflow:hidden;
            }
        </style>
    </head>
    <body>
        <div>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS). It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</div>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      div
      {
        width:200px;
        height:200px;
        border:2px solid black;
        overflow:scroll;
      }
    </style>
  </head>
  <body>
    <div>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS). It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
    </div>
  </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      div
      {
        width:200px;
        height:200px;
        border:2px solid black;
        overflow:auto;
      }
    </style>
  </head>
  <body>
    <div>
```

```

        }
    </style>
</head>
<body>
    <div>

```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites.

```
        </div>
    </body>
</html>
```

CSS box-shadow property

The box-shadow property attaches one or more shadows to an element.

Syntax box-shadow: none |h-offset v-offset blur spread color
 ex: box-shadow: 2px 2px 3px 10px blue;

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                width:200px;
                height:200px;
                margin:50px auto;
                border-radius:15px;
                overflow:hidden;
                box-shadow: 2px 2px 25px 6px #FF0000;
            }
        </style>
    </head>
    <body>
        <div>
        </div>
    </body>
</html>
```

ex:

```
-----
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                width:200px;

```

```

        height:200px;
        margin:50px auto;
        border-radius:15px;
        overflow:hidden;
        box-shadow: 2px 2px 25px 6px #FF0000 inset;
    }
</style>
</head>
<body>
<div>

</div>
</body>
</html>

```

HTML table creation

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
    <style>

    </style>
</head>
<body>
    <table border="1" text-align="center" width="100%">
        <thead>
            <tr>
                <th>No</th>
                <th>NAME</th>
                <th>ADDRESS</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>101</td>
                <td>Alan</td>
                <td>Texas</td>
            </tr>
            <tr>
                <td>102</td>
                <td>Jose</td>
                <td>Florida</td>
            </tr>
            <tr>
                <td>103</td>
                <td>Nancy</td>
                <td>Miami</td>
            </tr>
        </tbody>
    </table>

```

```

        </table>
    </body>
</html>

CSS Sample Design
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            body
            {
                background-image: url("images/bg.jpg");
                background-repeat:no-repeat;
                background-size:cover;
                background-position:center 0px;
                background-attachment:fixed;
            }
            div
            {
                width:400px;
                height:400px;
                box-shadow:2px 2px 20px 6px #FFF;
                background-color:transparent;
                margin:50px auto;
            }
            p
            {
                color:#FFF;
                font-weight:bold;
                padding:20px;
                text-align:justify;
            }
        </style>
    </head>
    <body>
        <div>
            <p>

```

Lesson Summary. Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

Lesson Summary. Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

Lesson Summary. Web technology refers to the means by which.

by which.

Lesson Summary. Web technology refers to the means

by which.

Lesson Summary. Web technology refers to the means

```
</p>
</div>
</body>
</html>
```

CSS float property

It is widely used property on a web page.

The float property specifies how an element should float.

<u>value</u>	<u>Description</u>
none	The element does not float.
left	the element floats to the left of its container.
right	The element floats to the right of its container.

ex:1

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
    <style>
        div
        {
            width:300px;
            height:300px;
            border:2px solid blue;
            float:none;
        }
    </style>
</head>
<body>
    <div>
        </div>
        <div>
            </div>
    </div>
</body>
</html>
```

ex:2

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
    <style>
        div
        {
            width:300px;
        }
    </style>

```

```
height:300px;  
border:2px solid blue;  
float:left;  
}  
</style>  
</head>  
<body>  
<div>  
  
</div>  
<div>  
  
</div>  
</body>  
</html>
```

ex:3

```
<!DOCTYPE html>  
<html>  
<head>  
<title>MyPage!</title>  
<style>  
div  
{  
width:300px;  
height:300px;  
border:2px solid blue;  
float:right;  
}  
</style>  
</head>  
<body>  
<div>  
  
</div>  
<div>  
  
</div>  
</body>  
</html>
```

ex:4

```
<!DOCTYPE html>  
<html>  
<head>  
<title>MyPage!</title>  
<style>  
div  
{
```

```

        width:300px;
        height:300px;
        border:2px solid blue;
        float:left;
        margin:50px 140px;
    }
</style>
</head>
<body>
<div>

</div>
<div>

</div>
</body>
</html>

```

Types of selectors in CSS

We have five selectors in CSS.

1) Element selector

The element selector selects HTML elements based on element name.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                color:blue;
                background-color:cyan;
                text-align:center;
            }
        </style>
    </head>
    <body>
        <h1> Heading Tag </h1>
        <h1> Heading Tag </h1>
    </body>
</html>

```

2) Id selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element is unique within a page, so the id selector is used to select one unique element.

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            #id1
            {
                color:blue;
                background-color:cyan;
            }
            #id2
            {
                text-align:center;
            }
        </style>
    </head>
    <body>
        <h1 id="id1"> Heading Tag </h1>
        <h1 id="id2"> Heading Tag </h1>
    </body>
</html>
```

Once element can accept only one ID at a time.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            #id1
            {
                color:blue;
                background-color:cyan;
            }
            #id2
            {
                text-align:center;
            }
        </style>
    </head>
    <body>
        <h1 id="id1"> Heading Tag </h1>
        <h1 id="id2"> Heading Tag </h1>
        <h1 id="id1 id2"> Heading Tag</h1>
    </body>
</html>
```

3) Class selector

The class selector selects HTML elements with a specific class attribute.
To select elements with a specific class, write a period (.) character, followed by the class name.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            .myClass1
            {
                color:blue;
                background-color:cyan;
            }
            .myClass2
            {
                text-align:center;
            }
        </style>
    </head>
    <body>
        <h1 class="myClass1"> Heading Tag </h1>
        <h1 class="myClass2"> Heading Tag </h1>
    </body>
</html>
```

We can declare multiple classes in a single element.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            .myClass1
            {
                color:blue;
                background-color:cyan;
            }
            .myClass2
            {
                text-align:center;
            }
        </style>
    </head>
    <body>
        <h1 class="myClass1"> Heading Tag </h1>
        <h1 class="myClass2"> Heading Tag </h1>
        <h1 class="myClass1 myClass2"> Heading Tag</h1>
    </body>
</html>
```

```
</body>
</html>
```

4) Group selector

The grouping selector selects all the HTML elements with the same style definitions

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1,p,div
            {
                color:red;
                background-color:rgb(255, 255, 0);
                text-align:center;
            }
        </style>
    </head>
    <body>
        <h1>Heading Tag</h1>
        <p>Paragraph Tag</p>
        <div>Division Tag</div>
    </body>
</html>
```

5) Universal selector

The universal selector (*) selects all HTML elements on the page.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            *
            {
                color:red;
                background-color:rgb(255, 255, 0);
                text-align:center;
            }
        </style>
    </head>
    <body>
        <h1>Heading Tag</h1>
        <p>Paragraph Tag</p>
        <div>Division Tag</div>
    </body>
</html>
```

CSS clear property

The clear property controls the flow next to floated elements.

ex:1

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            #box1
            {
                width:200px;
                height:200px;
                background-color:rgb(255,0,0);
                float:left;
            }
            #box2
            {
                width:200px;
                height:200px;
                background-color:hsl(60, 100%, 50%);
                float:left;
            }
        </style>
    </head>
    <body>
        <div id="box1"></div>
        <div id="box2"></div>
    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            #box1
            {
                width:200px;
                height:200px;
                background-color:rgb(255,0,0);
                float:left;
            }
            #box2
            {
                width:200px;
                height:200px;
                background-color:hsl(60, 100%, 50%);
                float:right;
            }
        </style>
    </head>
    <body>
        <div id="box1"></div>
        <div id="box2"></div>
    </body>
</html>
```

```
        </style>
    </head>
    <body>
        <div id="box1"></div>
        <div id="box2"></div>
    </body>
</html>
```

ex:3

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            #box1
            {
                width:200px;
                height:200px;
                background-color:rgb(255,0,0);
                float:left;
            }
            #box2
            {
                width:200px;
                height:200px;
                background-color:hsl(60, 100%, 50%);
                float:right;
            }
            #box3
            {
                width:100%;
                height:200px;
                background-color:#00ff00;
                clear:both;
            }
        </style>
    </head>
    <body>
        <div id="box1"></div>
        <div id="box2"></div>
        <div id="box3"></div>
    </body>
</html>
```

ex:4

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
```

```

#box1
{
    width:200px;
    height:200px;
    background-color:rgb(255,0,0);
    float:left;
}
#box2
{
    width:200px;
    height:200px;
    background-color:hsl(60, 100%, 50%);
    float:right;
}
#box3
{
    width:100%;
    height:200px;
    background-color:#00ff00;
    clear:both;
}
</style>
</head>
<body>
    <div id="box1"></div>
    <div id="box2"></div>
    <div id="box3"></div>
</body>
</html>

```

ex:5

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            #box1
            {
                width:200px;
                height:300px;
                background-color:rgb(255,0,0);
                float:left;
            }
            #box2
            {
                width:200px;
                height:200px;
                background-color:hsl(60, 100%, 50%);
                float:right;
            }
        </style>
    </head>
    <body>
        <div id="box1"></div>
        <div id="box2"></div>
    </body>
</html>

```

```
#box3
{
    width:100%;
    height:200px;
    background-color:#00ff00;
    clear:left;
}
</style>
</head>
<body>
    <div id="box1"></div>
    <div id="box2"></div>
    <div id="box3"></div>
</body>
</html>
```

ex:6

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            #box1
            {
                width:200px;
                height:300px;
                background-color:rgb(255,0,0);
                float:left;
            }
            #box2
            {
                width:200px;
                height:200px;
                background-color:hsl(60, 100%, 50%);
                float:right;
            }
            #box3
            {
                width:100%;
                height:200px;
                background-color:#00ff00;
                clear:right;
            }
        </style>
    </head>
    <body>
        <div id="box1"></div>
        <div id="box2"></div>
        <div id="box3"></div>
    </body>
```

```
</html>
```

CSS Display property

The display property specifies the display behavior of an element.

syntax : display: value;

<u>value</u>	<u>Description</u>
none	The element is completely removed.
inline	displays an element as on inline element. Any height and width properties will not effect.
block	Displays an element as block element. IT starts on a new line and takes up the whole width.
inline-block	Displays an element as an inline-level container. The element itself is formatting as an inline element. but we can apply height and width values.
inherit	Inherits this property from its parent element.

ex:1

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                display:none;
            }
            p
            {
                display:none;
            }
        </style>
    </head>
    <body>
        <h1>Heading Tag</h1>
        <p>Paragraph Tag</p>
    </body>
</html>
```

ex:2

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                display:inline;
            }
        </style>
    </head>
    <body>
```

```
<div>Division Tag</div>
<div>Division Tag</div>
<div>Division Tag</div>
</body>
</html>
```

ex:3

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            span
            {
                display:block;
                border:2px solid black;
                width:50%;
            }
        </style>
    </head>
    <body>
        <span>Span Tag</span>
        <span>Span Tag</span>
        <span>Span Tag</span>
    </body>
</html>
```

ex:4

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            span
            {
                display:inline-block;
                border:2px solid black;
                width:200px;
            }
        </style>
    </head>
    <body>
        <span>Span Tag</span>
        <span>Span Tag</span>
        <span>Span Tag</span>
    </body>
</html>
```

box-sizing property

The box-sizing property defines how the width and height of an element are calculated: should they include padding and borders, or not.

box-sizing: content-box

It will take separate width, border and padding. But it will not take margin.

ex:1

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                width:300px;
                height:300px;
                background-color:yellow;
                border:2px solid red;
                padding:10px;
                box-sizing:content-box;
            }
        </style>
    </head>
    <body>
        <div></div>
    </body>
</html>
```

box-sizing: border-box

It includes width, border and padding. But it will not include margin.

ex:2

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                width:300px;
                height:300px;
                background-color:yellow;
                border:2px solid red;
                padding:10px;
                box-sizing:border-box;
            }
        </style>
    </head>
    <body>  <div>      </div>  </body>
</html>
```

CSS Design 1

Task1

```
|  
|----images  
|      |  
|      |----rock.png  
|----css  
|      |  
|      |----mystyles.css  
|  
|----index.html
```

index.html

```
<!DOCTYPE html>  
<html>  
    <head>  
        <title>MyPage</title>  
  
        <!-- add external css -->  
        <link rel="stylesheet" href="css/mystyles.css">  
    </head>  
    <body>  
        <div class="container">  
            <div class="box1">  
                  
            </div>  
            <div class="box2">  
                <h3>Dwayne Johnson</h3>  
                <p>
```

Dwayne Douglas Johnson (born May 2, 1972), also known by his ring name the Rock, is an American actor and retired professional wrestler. Widely regarded as one of the greatest professional wrestlers of all time,[6][7] he was integral to the development and success of the World Wrestling Federation (WWF, now WWE) during the Attitude Era, an industry boom period in the late 1990s and early 2000s. Johnson wrestled for the WWF for eight years

```
                </p>  
                <p>
```

Dwayne Douglas Johnson (born May 2, 1972), also known by his ring name the Rock, is an American actor and retired professional wrestler. Widely regarded as one of the greatest professional wrestlers of all time,[6][7] he was integral to the development and success of the World Wrestling Federation (WWF, now WWE) illion worldwide,[8] making him one of the world's highest-grossing and highest-paid actors

```
                </p>  
                </div>  
            </div>  
        </body>  
</html>
```

```
mystyles.css
*
{
    margin:0;
    padding:0;
}
.container
{
    width:800px;
    height:400px;
    box-sizing:border-box;
    background-color: #34ace0;
    margin: 150px auto;
    box-shadow: 2px 2px 16px 6px #c3c3c3;
}
.box1
{
    width:400px;
    height:400px;
    box-sizing:border-box;
    background-color:#33d9b2;
    float:left;
}
.box1 img
{
    width:100%;
    height:100%;
}
.box2
{
    width:400px;
    height:400px;
    box-sizing:border-box;
    background-color:#f7f1e3;
    float:right;
    padding:20px;
}
.box2 h3
{
    text-align:center;
    padding:10px;
    text-transform:uppercase;
}
.box2 p
{
    text-align:justify;
}
```

CSS Design2

Task1

```
|  
|----css  
|      |  
|      |---mystyles.css  
|  
|---index.html
```

index.html

```
<!DOCTYPE html>  
<html>  
    <head>  
        <title>MyPage</title>  
  
        <!-- add external css -->  
        <link rel="stylesheet" href="css/mystyles.css">  
    </head>  
    <body>  
        <header class="header">  
            <h3>I<span>HUB</span>TALENT</h3>  
            <ul>  
                <li><a href="">Home</a></li>  
                <li><a href="">About</a></li>  
                <li><a href="">Service</a></li>  
                <li><a href="">Portfolio</a></li>  
                <li><a href="">Contact</a></li>  
            </ul>  
        </header>  
    </body>  
</html>
```

mystyles.css

```
*  
{  
    margin:0;  
    padding:0;  
}  
.header  
{  
    width:100%;  
    background-color:#3dc1d3;  
    padding:20px;  
}  
.header h3  
{  
    float:left;  
}  
.header ul  
{
```

```

        float:right;
        list-style-type:none;
    }
.header span
{
    font-style:italic;
    color:#FFF;
}
.header ul li
{
    display:inline-block;
    margin: 0px 25px;
    line-height:23px;
}
.header ul li a
{
    text-decoration:none;
    color:#000;
    font-size:13px;
    text-transform:uppercase;
}
.header ul li a:hover
{
    color:#FFF;
}

```

CSS transform property

CSS transform property allows use to move ,rotate or skew elements.

CSS transform property contains following transformation methods.

ex:

```

translate()
rotate()
scaleX()
scaleY()
skewX()
skewY()
skew()

```

transform: translate()

The translate() method moves an element from its current position to the parameters given by the X-axis and the Y-axis.

ex:

ex:1

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div

```

```

        {
            width:200px;
            height:200px;
            border:2px solid black;
            box-sizing:border-box;
            transform:translate(200px,100px);
        }
    </style>
</head>
<body>
    <div>
        </div>
    </body>
</html>

```

transform: scaleX()

The scaleX() method increases and decreases the width of the element.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>IHUB TALENT</title>
        <style>
            div
            {
                width:300px;
                height:200px;
                border:2px solid black;
                margin:100px auto;
                transform:scaleX(2);
            }
        </style>
    </head>
    <body>
        <div>
            </div>
        </body>
    </html>

```

HTML is widely used language on web to develop web pages and web applications

transform: scaleY()

The scaleY() method increases and decreases the height of the element.

ex:

```
<!DOCTYPE html>
```

```

<html>
  <head>
    <title>IHUB TALENT</title>
    <style>
      div
      {
        width:300px;
        height:200px;
        border:2px solid black;
        margin:100px auto;
        transform:scaleY(2);
      }
    </style>
  </head>
  <body>
    <div>
      HTML is widely used language on web to develop web pages and web
      applications
    </div>
  </body>
</html>

```

transform : scale()

The scaleX() method increases and decreases the width and height of the element.

ex:

```

<!DOCTYPE html>
<html>
  <head>
    <title>IHUB TALENT</title>
    <style>
      div
      {
        width:300px;
        height:200px;
        border:2px solid black;
        margin:100px auto;
        transform:scale(2);
      }
    </style>
  </head>
  <body>
    <div>
      HTML is widely used language on web to develop web pages and web
      applications
    </div>
  </body>
</html>

```

transform: skewX()

The skewX() method skews an element along the X-axis by the given angle.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB TALENT</title>
        <style>
            div
            {
                width:300px;
                height:200px;
                border:2px solid black;
                margin:100px auto;
                transform:skewX(30deg);
            }
        </style>
    </head>
    <body>
        <div>
            HTML is widely used language on web to develop web pages and web
            applications
        </div>
    </body>
</html>
```

transform: skewY()

The skewY() method skews an element along the Y-axis by the given angle.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB TALENT</title>
        <style>
            div
            {
                width:300px;
                height:200px;
                border:2px solid black;
                margin:100px auto;
                transform:skewY(30deg);
            }
        </style>
    </head>
    <body>
        <div>
            HTML is widely used language on web to develop web pages and web
            applications
        </div>
    </body>
</html>
```

```
</body>
</html>
```

transform: skew()

The skew() method skews an element along the X-axis and Y-axis by the given angle.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB TALENT</title>
        <style>
            div
            {
                width:300px;
                height:200px;
                border:2px solid black;
                margin:100px auto;
                transform:skew(30deg);
            }
        </style>
    </head>
    <body>
        <div>
            HTML is widely used language on web to develop web pages and web
            applications
        </div>
    </body>
</html>
```

transform: rotate()

The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

ex:8

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                width:200px;
                height:200px;
                border:2px solid black;
                box-sizing:border-box;
                margin:50px auto;
                transform:rotate(30deg);
            }
        </style>
    </head>
    <body>
        <div>
            HTML is widely used language on web to develop web pages and web
            applications
        </div>
    </body>
</html>
```

```

        </style>
</head>
<body>
    <div>
        </div>
    </body>
</html>

```

ex:9

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                width:200px;
                height:200px;
                border:2px solid black;
                box-sizing:border-box;
                margin:50px auto;
                transform:rotate(-30deg);

            }
        </style>
    </head>
    <body>
        <div>

```

CSS is used to apply the styles on elements to describe how an element should look like and to perform some sort of

animations.

```

            </div>
        </body>
</html>

```

CSS transition property

CSS transition property allows us to change property value smoothly , over a given duration.
To see the effects on an element ,we need to use mouse over to the element.

We have following properties in CSS transition.

ex:

- transition-delay
- transition-duration
- transition-property
- transition-timing-function
- or
- transition

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      div
      {
        width:200px;
        height:200px;
        background-color:red;
        box-sizing:border-box;
        transition-property:width,height;
      }
      div:hover
      {
        width:400px;
        height:400px;
      }
    </style>
  </head>
  <body>
    <div>
    </div>
  </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      div
      {
        width:200px;
        height:200px;
        background-color:red;
        box-sizing:border-box;
        transition-property:all;
      }
      div:hover
      {
        width:400px;
        height:400px;
      }
    </style>
  </head>
  <body>
    <div>
```

```
</div>
</body>
</html>

ex:
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                width:200px;
                height:200px;
                background-color:red;
                box-sizing:border-box;
                transition-property:all;
                transition-duration:3s;
            }
            div:hover
            {
                width:400px;
                height:400px;
            }
        </style>
    </head>
    <body>
        <div>
        </div>
    </body>
</html>
```

```
ex:
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                width:200px;
                height:200px;
                background-color:red;
                box-sizing:border-box;
                transition-property:all;
                transition-duration:3s;
                transition-timing-function:linear;
            }
            div:hover
            {
```

```
        width:400px;
        height:400px;
    }

```

```
</style>

```

```
</head>

```

```
<body>
    <div>
    </div>

```

```
</body>

```

```
</html>
```

ex:

```
<!DOCTYPE html>

```

```
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                width:200px;
                height:200px;
                background-color:red;
                box-sizing:border-box;
                transition-property:all;
                transition-duration:3s;
                transition-timing-function:ease-in;
            }
            div:hover
            {
                width:400px;
                height:400px;
            }
        </style>
    </head>
    <body>
        <div>
        </div>
    </body>

```

```
</html>
```

ex:

```
<!DOCTYPE html>

```

```
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                width:200px;
                height:200px;

```

```
        background-color:red;
        box-sizing:border-box;
        transition-property:all;
        transition-duration:3s;
        transition-timing-function:ease-out;
    }
    div:hover
    {
        width:400px;
        height:400px;
    }
</style>
</head>
<body>
    <div>
    </div>
</body>
</html>
```

ex:

```
<!DOCTYPE html>

<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                width:200px;
                height:200px;
                background-color:red;
                box-sizing:border-box;
                transition-property:all;
                transition-duration:3s;
                transition-timing-function:ease-in-out;
            }
            div:hover
            {
                width:400px;
                height:400px;
            }
        </style>
    </head>
    <body>
        <div>
        </div>
    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                width:200px;
                height:200px;
                background-color:red;
                box-sizing:border-box;
                transition-property:all;
                transition-duration:3s;
                transition-timing-function:ease-in-out;
                transition-delay:5s;
            }
            div:hover
            {
                width:400px;
                height:400px;
            }
        </style>
    </head>
    <body>
        <div>
        </div>
    </body>
</html>
```

ex:

```
<!DOCTYPE html>

<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                width:200px;
                height:200px;
                background-color:red;
                box-sizing:border-box;
                transition:all 2s linear 3s;
            }
            div:hover
            {
                width:400px;
                height:400px;
                background-color:blue;
            }
        </style>
    </head>
    <body>
        <div>
        </div>
    </body>
</html>
```

```
        }
    </style>
</head>
<body>
    <div>
        </div>
    </body>
</html>
```

CSS Design 1

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            img
            {
                width:250px;
                height:250px;
                margin:100px 400px;
                transition:all 2s linear;
            }
            img:hover
            {
                transform:rotate(360deg);
            }
        </style>
    </head>
    <body>
        
    </body>
</html>
```

CSS Design 2

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            img
            {
                width:300px;
                height:300px;
                margin:100px 30px;
                filter:grayscale(100%);
                transition:all 1s linear;
            }
            img:hover
            {
```

```

        transform:scale(1.2);
        filter:grayscale(0);
    }

```

</style>

</head>

<body>

<div>

</div>

</body>

</html>

CSS position property

The position property specifies the type of positioning method used for an element (static, relative, absolute, fixed, or sticky).

The following are the list of values to position property.

static

It is default value. Elements render in order, as they appear in the document flow.

absolute

The element is positioned relative to its first positioned (not static) ancestor element.

fixed

The element is positioned relative to the browser window.

relative

The element is positioned relative to its normal position, so "left:20px" adds 20 pixels to the element's LEFT position.

sticky

The element is positioned based on the user's scroll position

css position property mandatory should have following properties.

- i) left
- ii) right
- iii) top
- iv) bottom

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            img
            {
                width:200px;
                height:200px;
                border:2px solid black;
                position:static;
                top:0;
                left:0;
            }
        </style>
    </head>
    <body>
        
    </body>
</html>

```

```
        }
    </style>
</head>
<body>
    <p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
    </p>
    <p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
    </p>
    
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
    </p>
    <p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
    </p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
    </p>
    <p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
    </p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
    </p>
```

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

</body>

</html>

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      img
      {
        width:200px;
        height:200px;
        border:2px solid black;
        position:absolute;
        top:100px;
        left:100px;
      }
    </style>
  </head>
  <body>
    <h1>Hello World!</h1>
    <img alt="A small square image." data-bbox="100 100 200 200"/>
  </body>
</html>
```

```
</style>
</head>
<body>
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>

```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

</body>

</html>

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            img
            {
                width:200px;
                height:200px;
                border:2px solid black;
                position:relative;
                top:100px;
                left:100px;
            }
        </style>
    </head>
    <body>
        <h1>Hello World!</h1>
        
    </body>
</html>
```

```
</head>
<body>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```

```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

</body>

</html>

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      img
      {
        width:200px;
        height:200px;
        border:2px solid black;
        position:fixed;
        top:100px;
        left:100px;
      }
    </style>
```

```
</head>
<body>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```

```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

</body>

</html>

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      img
      {
        width:200px;
        height:200px;
        border:2px solid black;
        position:sticky;
        top:100px;
        left:100px;
      }
    </style>
```

```
</head>
<body>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```

```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

</body>

</html>

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage</title>
    <style>

      .myClass1
      {
        width:200px;
        height:200px;
        position:relative;
        left:420px;

      }

      .myClass2
```

```

        {
            width:200px;
            height:200px;
        }

    </style>
</head>
<body>
    
    
</body>
</html>

```

CSS z-index property

The z-index property specifies the stack order of an element.

An element with greater stack order is always in front of an element with a lower stack order.

Note: z-index only works on positioned elements.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            img
            {
                width:200px;
                height:200px;
                border:2px solid black;
                background-color:#FFF;
                position:absolute;
                top:0;
                left:0;
                z-index:1;
            }
        </style>
    </head>
    <body>

```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

<p>

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

</p>

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```

```

```
</body>  
</html>
```

ex:

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>MyPage!</title>  
    <style>  
      img  
      {  
        width:200px;  
        height:200px;  
        border:2px solid black;  
        background-color:#FFF;  
        position:absolute;  
        top:0;  
        left:0;  
        z-index:-1;  
      }  
    </style>  
  </head>  
  <body>  
    <p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```
<p>
```

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS).

```
</p>
```

```

```

```
</body>
```

```
</html>
```

CSS opacity property

The opacity property sets the opacity level for an element.

The opacity-level describes the transparency-level, where 1 is not transparent at all, 0.5 is 50% see-through, and 0 is completely transparent.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                margin:50px auto;
                background-color:red;
                width:200px;
                height:200px;
                opacity:1;
            }
        </style>
    </head>
    <body>
        <div>
        </div>
    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
```

```

        {
            margin:50px auto;
            background-color:red;
            width:200px;
            height:200px;
            opacity:0.5;
        }
    </style>
</head>
<body>
    <div>

    </div>
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            div
            {
                margin:50px auto;
                background-color:red;
                width:200px;
                height:200px;
                opacity:0.1;
            }
        </style>
    </head>
    <body>
        <div>

        </div>

    </body>
</html>

```

CSS Gradients

CSS gradients let you display smooth transitions between two or more specified colors.
CSS defines three types of gradients:

- 1) Linear Gradients (goes down/up/left/right/diagonally)
- 2) Conic Gradients (rotated around a center point)
- 3) Radial Gradients (defined by their center)

ex:

```

<!DOCTYPE html>
<html>

```

```
<head>
    <title>MyPage!</title>
    <style>
        body
        {
            height:100vh;
            background:linear-gradient(yellow,red);
        }
    </style>
</head>
<body>

</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            body
            {
                height:100vh;
                background:linear-gradient(to left,yellow,red);
            }
        </style>
    </head>
    <body>

    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            body
            {
                height:100vh;
                background-image:conic-gradient(yellow,red);
            }
        </style>
    </head>
    <body>

    </body>
</html>
```

```

ex:
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            body
            {
                height:100vh;
                background-image:radial-gradient(yellow,red);
            }
        </style>
    </head>
    <body>

    </body>
</html>

```

CSS flexbox

It is a one-dimensional layout methods for laying out our items in rows and columns.
 CSS flexbox is a better way to align items into a container.

Flexbox= flexible + box.

To create a flexbox model, we need to define a "flex container".

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            .container
            {
                width:100%;
                height:400px;
                border:2px solid black;
                display:flex;
                flex-direction:row;
            }
            .container .item
            {
                width:150px;
                height:150px;
                border:2px solid black;
            }
        </style>
    </head>
    <body>
        <div class="container">
            <div class="item">Item1</div>
            <div class="item">Item2</div>

```

```
<div class="item">Item3</div>
<div class="item">Item4</div>
<div class="item">Item5</div>
<div class="item">Item6</div>
</div>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            .container
            {
                width:100%;
                height:400px;
                border:2px solid black;
                display:flex;
                flex-direction:row-reverse;
            }
            .container .item
            {
                width:150px;
                height:150px;
                border:2px solid black;
            }
        </style>
    </head>
    <body>
        <div class="container">
            <div class="item">Item1</div>
            <div class="item">Item2</div>
            <div class="item">Item3</div>
            <div class="item">Item4</div>
            <div class="item">Item5</div>
            <div class="item">Item6</div>
        </div>
    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            .container
```

```

    {
        width:100%;
        height:400px;
        border:2px solid black;
        display:flex;
        flex-direction:column;
    }
    .container .item
    {
        width:150px;
        height:150px;
        border:2px solid black;
    }

```

</style>

</head>

<body>

```

<div class="container">
    <div class="item">Item1</div>
    <div class="item">Item2</div>
    <div class="item">Item3</div>
    <div class="item">Item4</div>
    <div class="item">Item5</div>
    <div class="item">Item6</div>
</div>

```

</body>

</html>

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            .container
            {
                width:100%;
                height:400px;
                border:2px solid black;
                display:flex;
                flex-direction:column-reverse;
            }
            .container .item
            {
                width:150px;
                height:150px;
                border:2px solid black;
            }
        </style>
    </head>

```

```
<body>
<div class="container">
    <div class="item">Item1</div>
    <div class="item">Item2</div>
    <div class="item">Item3</div>
    <div class="item">Item4</div>
    <div class="item">Item5</div>
    <div class="item">Item6</div>
</div>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            .container
            {
                width:100%;
                height:400px;
                border:2px solid black;
                display:flex;
                flex-direction:row;
                flex-wrap:no-wrap;
            }
            .container .item
            {
                width:150px;
                height:150px;
                border:2px solid black;
            }
        </style>
    </head>
    <body>
        <div class="container">
            <div class="item">Item1</div>
            <div class="item">Item2</div>
            <div class="item">Item3</div>
            <div class="item">Item4</div>
            <div class="item">Item5</div>
            <div class="item">Item6</div>
        </div>
    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      .container
      {
        width:100%;
        height:400px;
        border:2px solid black;
        display:flex;
        flex-direction:row;
        flex-wrap:wrap;
      }
      .container .item
      {
        width:150px;
        height:150px;
        border:2px solid black;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <div class="item">Item1</div>
      <div class="item">Item2</div>
      <div class="item">Item3</div>
      <div class="item">Item4</div>
      <div class="item">Item5</div>
      <div class="item">Item6</div>
    </div>
  </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      .container
      {
        width:100%;
        height:400px;
        border:2px solid black;
        display:flex;
        flex-flow:row wrap;
      }
    </style>
  </head>
  <body>
    <div class="container">
```

```

.container .item
{
    width:150px;
    height:150px;
    border:2px solid black;
}

</style>
</head>
<body>
<div class="container">
    <div class="item">Item1</div>
    <div class="item">Item2</div>
    <div class="item">Item3</div>
    <div class="item">Item4</div>
    <div class="item">Item5</div>
    <div class="item">Item6</div>
</div>
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            .container
            {
                width:100%;
                height:400px;
                border:2px solid black;
                display:flex;
                flex-flow:row wrap;
                justify-content:center;
            }
            .container .item
            {
                width:150px;
                height:150px;
                border:2px solid black;
            }
        </style>
    </head>
    <body>
        <div class="container">
            <div class="item">Item1</div>
            <div class="item">Item2</div>
            <div class="item">Item3</div>

```

```
<div class="item">Item4</div>
<div class="item">Item5</div>
<div class="item">Item6</div>
</div>
</body>
</html>

ex:
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            .container
            {
                width:100%;
                height:400px;
                border:2px solid black;
                display:flex;
                flex-flow:row wrap;
                justify-content:center;
                align-items:center;
            }
            .container .item
            {
                width:150px;
                height:150px;
                border:2px solid black;
            }
        </style>
    </head>
    <body>
        <div class="container">
            <div class="item">Item1</div>
            <div class="item">Item2</div>
            <div class="item">Item3</div>
            <div class="item">Item4</div>
            <div class="item">Item5</div>
            <div class="item">Item6</div>
        </div>
    </body>
</html>
```

```
ex:
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
```

```

.container
{
    width:100%;
    height:400px;
    border:2px solid black;
    display:flex;
    flex-flow:row wrap;
    align-content:center;
}
.container .item
{
    width:150px;
    height:150px;
    border:2px solid black;
}

</style>
</head>
<body>
<div class="container">
    <div class="item">Item1</div>
    <div class="item">Item2</div>
    <div class="item">Item3</div>
    <div class="item">Item4</div>
    <div class="item">Item5</div>
    <div class="item">Item6</div>
</div>
</body>
</html>

```

CSS Grid layout

The CSS grid layout module offers a grid-based layout system with rows and columns. Grid layout makes easier to design web pages without having a use of floats and positioning tag.

A grid layout consists of a parent element , with one or more child elements.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            .container
            {
                width:100%;
                height:400px;
                border:2px solid black;
                display:grid;
            }
            .item1{background-color:#EE5A24;}

```

```

.item2{background-color:#12CBC4;}
.item3{background-color:#C4E538;}
.item4{background-color:#B53471;}
.item5{background-color:#0652DD;}
.item6{background-color:#6F1E51;}

</style>
</head>
<body>
<div class="container">
<div class="item1">Item1</div>
<div class="item2">Item2</div>
<div class="item3">Item3</div>
<div class="item4">Item4</div>
<div class="item5">Item5</div>
<div class="item6">Item6</div>
</div>
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
    <style>
        .container
        {
            width:100%;
            height:400px;
            border:2px solid black;
            display:grid;
            grid-template-rows:150px 150px;
            grid-template-columns:150px 150px 150px;
        }
        .item1{background-color:#EE5A24;}
        .item2{background-color:#12CBC4;}
        .item3{background-color:#C4E538;}
        .item4{background-color:#B53471;}
        .item5{background-color:#0652DD;}
        .item6{background-color:#6F1E51;}

    </style>
</head>
<body>
<div class="container">
<div class="item1">Item1</div>
<div class="item2">Item2</div>
<div class="item3">Item3</div>
<div class="item4">Item4</div>

```

```

        <div class="item5">Item5</div>
        <div class="item6">Item6</div>
    </div>
</body>
</html>

ex:
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            .container
            {
                width:100%;
                height:400px;
                border:2px solid black;
                display:grid;
                grid-template-rows:150px 150px;
                grid-template-columns:150px 150px 1fr;
            }
            .item1{background-color:#EE5A24;}
            .item2{background-color:#12CBC4;}
            .item3{background-color:#C4E538;}
            .item4{background-color:#B53471;}
            .item5{background-color:#0652DD;}
            .item6{background-color:#6F1E51;}
        </style>
    </head>
    <body>
        <div class="container">
            <div class="item1">Item1</div>
            <div class="item2">Item2</div>
            <div class="item3">Item3</div>
            <div class="item4">Item4</div>
            <div class="item5">Item5</div>
            <div class="item6">Item6</div>
        </div>
    </body>
</html>
```

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            .container
            {
```

```

        width:100%;
        height:400px;
        border:2px solid black;
        display:grid;
        grid-template-rows:repeat(2,150px);
        grid-template-columns:repeat(3,1fr);
    }
.item1{background-color:#EE5A24;}
.item2{background-color:#12CBC4;}
.item3{background-color:#C4E538;}
.item4{background-color:#B53471;}
.item5{background-color:#0652DD;}
.item6{background-color:#6F1E51;}

</style>
</head>
<body>
<div class="container">
    <div class="item1">Item1</div>
    <div class="item2">Item2</div>
    <div class="item3">Item3</div>
    <div class="item4">Item4</div>
    <div class="item5">Item5</div>
    <div class="item6">Item6</div>
</div>
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
    <style>
        .container
        {
            width:100%;
            height:400px;
            border:2px solid black;
            display:grid;
            grid-template-rows:repeat(2,150px);
            grid-template-columns:repeat(3,1fr);
            grid-row-gap:10px;
            grid-column-gap:10px;
        }
        .item1{background-color:#EE5A24;}
        .item2{background-color:#12CBC4;}
        .item3{background-color:#C4E538;}
        .item4{background-color:#B53471;}
        .item5{background-color:#0652DD;}
    </style>
</head>
<body>
<div class="container">
    <div class="item1">Item1</div>
    <div class="item2">Item2</div>
    <div class="item3">Item3</div>
    <div class="item4">Item4</div>
    <div class="item5">Item5</div>
</div>
</body>
</html>

```

```

.item6{background-color:#6F1E51;}

</style>
</head>
<body>
<div class="container">
    <div class="item1">Item1</div>
    <div class="item2">Item2</div>
    <div class="item3">Item3</div>
    <div class="item4">Item4</div>
    <div class="item5">Item5</div>
    <div class="item6">Item6</div>
</div>
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
    <style>
        .container
        {
            width:100%;
            height:400px;
            border:2px solid black;
            display:grid;
            grid-template-rows:repeat(2,150px);
            grid-template-columns:repeat(3,1fr);
            grid-gap:20px;
        }
        .item1{background-color:#EE5A24;}
        .item2{background-color:#12CBC4;}
        .item3{background-color:#C4E538;}
        .item4{background-color:#B53471;}
        .item5{background-color:#0652DD;}
        .item6{background-color:#6F1E51;}
    </style>
</head>
<body>
<div class="container">
    <div class="item1">Item1</div>
    <div class="item2">Item2</div>
    <div class="item3">Item3</div>
    <div class="item4">Item4</div>
    <div class="item5">Item5</div>
    <div class="item6">Item6</div>
</div>

```

```
</body>
</html>
```

CSS Google Fonts

If we do not want to use any of the standard fonts in HTML, you can use Google Fonts.
Google Fonts are free to use, and have more than 1000 fonts to choose.

To use any google fonts we need to use below url.

ex: https://fonts.googleapis.com/

ex:1

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>

        <style>
            @import
url('https://fonts.googleapis.com/css2?family=Ultra&display=swap');
            h1
            {
                font-family: 'Ultra', serif;
            }
        </style>
    </head>
    <body>
        <h1>This is Heading Tag</h1>
    </body>
</html>
```

ex:2

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <link href="https://fonts.googleapis.com/css2?family=Ultra&display=swap"
rel="stylesheet">
        <style>

            h1
            {
                font-family: 'Ultra', serif;
            }
        </style>
    </head>
    <body>
        <h1>This is Heading Tag</h1>
    </body>
</html>
```

FontAwesome Icons

If we want to take icons in a CSS then we need to use fontawesome icons.

All fontawesome icons are font based.

To work with fontawesome icons we need to add below CDN link.

ex:

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
```

ex:1

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    </head>
    <body>
        <i class="fa fa-heart"></i>
        <i class="fa fa-home"></i>
        <i class="fa fa-phone"></i>
    </body>
</html>
```

ex:2

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    </head>
    <body>
        <i class="fa fa-heart" style="font-size:30px;color:red;"></i>
        <i class="fa fa-home" style="font-size:30px;color:blue;"></i>
        <i class="fa fa-phone" style="font-size:30px;color:green;"></i>
    </body>
</html>
```

ex:3

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    </head>
    <body>
        <i class="fa fa-facebook" style="color:blue;"></i>
```

```

<i class="fa fa-instagram" style="color:pink;"></i>
<i class="fa fa-twitter" style="color:lightblue;"></i>
<i class="fa fa-youtube" style="color:Red;"></i>
<i class="fa fa-linkedin" style="color:blue;"></i>
</body>
</html>

```

CSS Cursor property

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                text-align:center;
            }
            h1:hover
            {
                cursor:pointer;
            }
        </style>
    </head>
    <body>
        <h1>Mouse Over Here </h1>
    </body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <style>
            h1
            {
                text-align:center;
            }
            h1:hover
            {
                cursor:zoom-in;
            }
        </style>
    </head>
    <body>
        <h1>Mouse Over Here </h1>
    </body>
</html>

```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      h1
      {
        text-align:center;
      }
      h1:hover
      {
        cursor:zoom-out;
      }
    </style>
  </head>
  <body>
    <h1>Mouse Over Here </h1>
  </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      h1
      {
        text-align:center;
      }
      h1:hover
      {
        cursor:progress;
      }
    </style>
  </head>
  <body>
    <h1>Mouse Over Here </h1>
  </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <style>
      h1
      {
```

```

        text-align:center;
    }
    h1:hover
    {
        cursor:copy;
    }
</style>
</head>
<body>
    <h1>Mouse Over Here </h1>
</body>
</html>

```

CSS Form

Form is used to accept the values from the enduser.
It will send the data to server or database for processing.

ex:

Task3

```

|
|----css
|    |
|    |---mystyles.css
|----images
|    |
|    |---micky.png
|
|----index.html

```

index.html

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage</title>
        <!-- add external css -->
        <link rel="stylesheet" href="css/mystyles.css">
    </head>
    <body>
        <div class="container">
            <div class="box1">
                
            <form action="">
                <input type="text" name="t1" autocomplete="off" placeholder="username" required/> <br>
                <input type="password" name="t2" placeholder="password" required/> <br>
                <input type="submit" value="Login"/>
            </form>
        </div>
    </body>
</html>

```

```
</div>
<div class="box2">
    
</div>
</div>
</body>
</html>
```

mystyles.css

```
*
{
    margin:0;
    padding:0;
}
body
{
    height:100vh;
    display:flex;
    justify-content:center;
    align-items:center;
    background: linear-gradient(yellow,red);
}
.container
{
    width:800px;
    height:400px;
    box-sizing:border-box;
}
.container .box1
{
    width:400px;
    height:400px;
    box-sizing:border-box;
    float:left;
    box-shadow:2px 2px 12px 6px #FFF;
}
.container .box1 img
{
    width:100px;
    height:100px;
    display:block;
    margin:20px auto;
    border-radius:50%;
    border:1px solid black;
}
.container .box1 input[type="text"],input[type="password"]
{
    width:300px;
    height:35px;
```

```
display:block;
margin:10px auto;
border:2px solid #FFF;
background-color:transparent;
border-radius:15px;
}
.container .box1 input[placeholder]
{
    text-align:center;
    color:#FFF;
    font-size:20px;
}

.container .box1 input[type="submit"]
{
    display:block;
    margin:10px auto;
    width:100px;
    padding:10px;
    border:none;
    font-family: cursive;
    font-size:15px;
    background-color:#FFF;
    transition:1s all linear;
}
.container .box1 input[type="submit"]:hover
{
    width:200px;
    border-radius:25px;
    border:1px solid #FFF;
    background:none;
    color:#FFF;
    cursor:pointer;
}

.container .box2
{
    width:400px;
    height:400px;
    box-sizing:border-box;
    float:right;
    box-shadow:2px 2px 12px 6px #FFF;
}
```

Java Script

Q) What is difference between Java and JavaScript?

<u>Java</u>	<u>JavaScript</u>
It is a non-scripting language.	It is a scripting language.
It is a object oriented programming language.	It is a object based programming language.
It is strongly typed checking language.	It is a weakly typed checking language.
It does not required browser window for execution.	It requires browser window for execution.
We can run individually.	We can't run individually.
It is a complex language.	It is easy language.

History of JavaScript

Original name of javascript is LiveScript.

LiveScript is developed by Netscape Corporation in late 1990's."

LiveScript was developed by using C language syntax's.

In 1995, Brenden Eich the popular scientist of Netscape Corporation renamed LiveScript to JavaScript.

Official name of JavaScript is ECMA script.

Here ECMA stands for European Computer Manufacturers Association.

Advantages of JavaScript

- 1) It is used to create interactive web pages.
- 2) It is used to display dialog boxes and popup boxes.
- 3) It is used to perform client side form validation.
- 4) It is used to add dynamic content to a web page.
- 5) It is used to develop responsive(dynamic) web pages.
- 6) It supports objects like String, Arrays, RegEx and etc.
- 7) It supports Date and Time.
- 8) It supports Cookies.
- 9) It supports dropdown menu. and etc.

Javascript syntax

To write javascript code we need to use <script> tag.

ex: <script type="text/javascript">
 stmt1;
 stmt2;
 stmt3;
 </script>

Here "type" attribute is optional.

Here simicolon(;) is optional.

or

ex: <script >
 stmt1
 stmt2
 stmt3
 </script>

Sublime Editor

Download link: <https://www.sublimetext.com/download>

```
ex:1
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script type="text/javascript">
        document.write("Welcome to JavaScript");
    </script>
</body>
</html>

ex:2
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        document.write("Welcome to JavaScript class")
    </script>
</body>
</html>

ex:3
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        document.writeln("Welcome to JavaScript class");
        document.writeln("This is Ihub Talent");
    </script>
</body>
</html>

ex:
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        document.writeln("Welcome to JavaScript class");
        <br>
```

```
        document.writeln("This is Ihub Talent");
    </script>
</body>
</html>
```

Note: We can't mix markup language in scripting language.

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        document.writeln("Welcome to JavaScript class");
        document.writeln("<br>");
        document.writeln("This is Ihub Talent");
    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        document.writeln("<h1 style='color:red'>Welcome to JavaScript class</h1>");
        document.writeln("<br>");
        document.writeln("<p style='color:blue'>This is Ihub Talent</p>");
    </script>
</body>
</html>
```

Note: If a program contains HTML, CSS and JavaScript then it is called DHTML.

DHTML stands for Dynamic Hypertext Markup language.

JavaScript Engine

JavaScript engine allows us to execute script code on browser window.

It converts user understandable scripting language to machine understandable scripting language.

By default , every browser contains javascript engine so we don't need to arrange it seperately.

We have following list of javascript engines.

ex:	<u>Browser</u>	<u>JavaScript Engine</u>
	Chrome	V8 Engine
	Mozilla	Spider monkey
	Edge	Chakra

and etc.

Comments in Javascript

Comments are created for documentation purpose.

Comments are used to improve readability of our code.

In javascript comments are divided into two types.

1) Single line comment

It is used to comment a single line.

ex: // comment here

2) Multiple line comment

It is more convenient because we can comment single line and multiple lines.

ex: /*

```
-  
- comment here  
-  
*/
```

Note: HTML -- <!-- comment here -->
CSS -- /* comment here */

Output statement in javascript

Whenever we want to display any userdefined statements or data then we need to use output statement.

We have following output statements in javascript.

- 1) document.writeln()
- 2) console.log()

1) document.writeln()

It is used to display the output on browser window.

ex:

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>MyPage!</title>  
</head>  
<body>  
    <script>  
        document.writeln("Hello World");  
    </script>  
</body>  
</html>
```

2) console.log()

It is used to display the output on browser console.

In order to see browser console we need to press F12 function key.

ex:

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>MyPage!</title>  
</head>  
<body>  
    <script>    console.log("Hello World");    </script>  
</body>  
</html>
```

JavaScript variables

A name which is given to a memory location is called variable.

Purpose of variable is used to store the data.

Javascript variables contains same rules as we have for identifiers.

Rule1:

A variable can starts with alphabet, underscore or dollar symbol but not with digit.

ex:

```
var abcd; //valid  
var _abcd; //valid  
var $abcd; //valid  
var a1234; //valid  
var 1abcd; //invalid
```

Rule2:

Every variable is a case sensitive.

ex:

```
var number;  
var NUMBER;  
var NumBer;
```

In javascript, we have two types of variables.

- 1) Local variable
- 2) Global variable

1) Local variable

A variable which is declared inside the function or block is called local variable.

A local variable we can access within the function but not outside the function.

ex:

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>MyPage!</title>  
</head>  
<body>  
    <script>  
        function f1()  
        {  
            //local variable  
            var i=10;  
  
            document.writeln(i+"<br>");  
        }  
  
        //calling  
        f1();  
    </script>  
</body>  
</html>
```

```

ex:2
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>

        function f1()
        {
            //local variable
            var i=10;
            document.writeln(i+"<br>");
        }
        function f2()
        {
            document.writeln(i+"<br>"); //Error
        }

        //calling
        f1();
        f2();

    </script>
</body>
</html>

```

2) Global variable

A variable which is declare outside the function or a block is called global variable.
A global variable we can access within the function and outside the function.

ex:1

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>

        //global variable
        var i=20;
        function f1()
        {
            document.writeln(i+"<br>");
        }
        //calling
        f1();

    </script>
</body>
</html>

```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        //global variable
        var i=20;
        function f1()
        {
            document.writeln(i+"<br>");
        }
        function f2()
        {
            document.writeln(i+"<br>");
        }
        //calling
        f1();
        f2();
    </script>
</body>
</html>
```

Note: One function can refer to another function.

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        function f1()
        {
            document.writeln("F1-Function1 <br>");
            f2();
        }
        function f2()
        {
            document.writeln("F1-Function2 <br>");
        }
        //calling
        f1();
    </script>
</body>
</html>
```

Datatypes

Javascript is a dynamically typed language it means we don't need to define datatype at the time of variable declaration.

To declare a variable we will use var keyword.

Javascript internally uses javascript engine to determine particular datatype based on the value.

In Javascript, we have two types datatypes.

- 1) Primitive datatypes
- 2) Non-Primitive datatypes

1) Primitive datatypes

We have following list of primitive datatypes.

Datatype	Description
1) number	It is used to represent numbers
2) boolean	It is used to represent boolean values.
3) String	It is used to represent string.
4) null	It is used to represent null.
5) undefined	It is used to represent undefined.

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i=10;
        document.writeln(i+"<br>");//10

        var j=10.5;
        document.writeln(j+"<br>");//10.5
    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i=true;
        document.writeln(i+"<br>");//true

        var j=false;
        document.writeln(j+"<br>");//false
    </script>

```

```

</body>
</html>

ex:
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i="hi";
        document.writeln(i+"<br>");//hi

        var j='bye';
        document.writeln(j+"<br>");//bye
    </script>
</body>
</html>

```

```

ex:
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i=null;
        document.writeln(i+"<br>");//null

        var j;
        document.writeln(j+"<br>");//undefined
    </script>
</body>
</html>

```

2) Non-Primitive datatypes

We have following list of non-primitive datatypes.

Datatype	Description
Object	It is used to create an instance through which we can access the members.
Arrays	It is used to represent similar elements.
RegEx	It is used to represent regular expression.

Types of javascript

We have two types of javascript.

- 1) Internal javascript / Embedded javascript
- 2) External javascript / Separate javascript

1) Internal javascript

In internal javascript we will maintain html code and javascript code in .html file.

Advantages:

There is no confusion of multiple files.

We can maintain HTML code and javascript code separately.

Disadvantages:

If code increases then it will increase complexity.

2) External javascript

In external javascript we will maintain HTML code in .html file and javascript code in .js file.
But we can't execute .js file separately in a browser window.

Advantages:

We can maintain html code in .html file and javascript code in .js file.

If code increases then it will not increase complexity.

Disadvantage:

Confusion of multiple files.

Operators

Operator is a symbol which is used to perform some operations on operands.

ex: c = a + b;

Here a , b and c are operands.

Here = and + are operators.

It can perform arithmetic operations, logical operations, conditional operations, assignment operators and etc.

We have following list of operators present in javascript.

- 1) Assignment operators
- 2) Conditional operators
- 3) Bitwise operators
- 4) Logical operators
- 5) Relational operators
- 6) Special operators

1) Assignment operators

We have following list of assignment operators.

Operator	Description
=	equals to
+=	addition and equals to
-=	subtraction and equals to
*=	multiplication and equals to
/=	division and equals to
%=	modules and equals to
++	incrementation
--	decremenataion

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
```

```
<script>
    var i=10;
    document.writeln(i+"<br>");//10

    var j=10;
    j+=5;
    document.writeln(j+"<br>");//15

    var k=10;
    k-=5;
    document.writeln(k+"<br>");//5

</script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i=10;
        i/=2;
        document.writeln(i+"<br>");//5

        var j=10;
        j/=20;
        document.writeln(j+"<br>");//0.5

    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i=10;
        i%=2;
        document.writeln(i+"<br>");//0

        var j=10;
        j%=20;
```

```

document.writeln(j+"<br>");//10

        </script>
</body>
</html>

ex:
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i=10;
        document.writeln(i++ + ++i+"<br>");//10 + 12 = 22

        var j=10;
        document.writeln(j-- - -j+"<br>");//10-8 =2

    </script>
</body>
</html>

ex:
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i=10;
        var j=i++ + i++ - -i - --i;
        document.writeln(i+ " "+j);//10  0

    </script>
</body>
</html>

```

2) Conditional operators

We have following list of conditional operators.

operator	description
>	greater than
>=	greater than equal to
<	less than
<=	less than equals to
==	equals to
!=	not equals to

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        document.writeln((10>20)+"<br>");//false
        document.writeln((10>=10)+"<br>");//true
        document.writeln((10<20)+"<br>");//true
        document.writeln((10<=20)+"<br>");//true
        document.writeln((10==20)+"<br>");//false
        document.writeln((10==10)+"<br>");//true
        document.writeln((10!=20)+"<br>");//true
        document.writeln((10!=10)+"<br>");//false
    </script>
</body>
</html>
```

Q)What is the difference between == and === ?

==

It is used to check values are same or not.

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        document.writeln((1 == 1)+"<br>");//true
        document.writeln((1 == true)+"<br>");//true
        document.writeln((0 == false)+"<br>");//true
        document.writeln(("1" == 1)+"<br>");//true
    </script>
</body>
</html>
```

== =

It is used to check values and datatypes are same or not.

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        document.writeln((1 === 1)+"<br>");//true
    </script>

```

```

        document.writeln((1 === true)+"<br>");//false
        document.writeln((0 === false)+"<br>");//false
        document.writeln(("1" === 1)+"<br>");//false
    </script>
</body>
</html>

```

3) Assignment operators

We have following list of assignment operators.

operator	description
<code>+=</code>	addition and equals to
<code>-=</code>	subtraction and equals to
<code>*=</code>	multiplication and equals to
<code>/=</code>	division and equals to
<code>%=</code>	modules and equals to
<code>==</code>	equals to
<code>!=</code>	not equals to

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i=10;
        i+=2;
        document.writeln(i); // 12
    </script>
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i=10;
        i%=2;
        document.writeln(i); // 0
    </script>
</body>
</html>

```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i=10;
        i%=20;
        document.writeln(i); // 10
    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i=10;
        i/=20;
        document.writeln(i); // 0.5
    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i=10;
        i*=20;
        document.writeln(i); // 200
    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
```

```

</head>
<body>
    <script>
        document.writeln((10 == 10) +"  
"); // true
        document.writeln((10 == 20) +"  
"); // false
        document.writeln((10 != 20) +"  
"); // true
        document.writeln((10 != 10) +"  
"); // false
    </script>
</body>
</html>

```

4) Logical operators

We have following list of logical operators.

operator	description
&&	logical AND
	logical OR
!	logical NOT

Logical AND operator (&&)

Truth table

T	T	= T
T	F	= F
F	T	= F
F	F	= F

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        document.writeln(((5>2) && (6<10)) +"  
"); // true
        document.writeln(((5>20) && (6<1)) +"  
"); // false
        document.writeln(((5>20) && (6<10)) +"  
"); // false
    </script>
</body>
</html>

```

Logical OR operator (||)

Truth table

T	T	= T
T	F	= T
F	T	= T
F	F	= F

ex:

```

<!DOCTYPE html>
<html>

```

```

<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        document.writeln(((5>2) || (6<10)) +"  
"); // true
        document.writeln(((5>20) || (6<1)) +"  
"); // false
        document.writeln(((5>20) || (6<10)) +"  
"); // true
    </script>
</body>
</html>

```

Logical NOT operator (!)

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        document.writeln((!(5>2)) +"  
"); // false
        document.writeln((!(5>20)) +"  
"); // true
    </script>
</body>
</html>

```

5) Bitwise Operators

We have following list of bitwise operators.

ex:

operator	description
&	Bitwise AND
	Bitwise OR
^	Bitwise XOR
~	Bitwise NOT
>>	Right shift
<<	left shift

Bitwise AND (&)

Bitwise AND operator deals with binary numbers.

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var a=10, b=5;
        var c = a & b;
    </script>

```

```

        document.writeln(c); //0
        /*
          10 - 1010
          5 - 0101
          -----
          & - 0000
        */
      </script>
    </body>
  </html>

```

Bitwise OR

Bitwise OR operator deals with binary numbers.

ex:

```

<!DOCTYPE html>
<html>
<head>
  <title>MyPage!</title>
</head>
<body>
  <script>
    var a=10, b=5;
    var c = a | b;
    document.writeln(c); //15
    /*
      10 - 1010
      5 - 0101
      -----
      | - 1111
    */
  </script>
</body>
</html>

```

Bitwise XOR operator (^)

Bitwise XOR operator deals with binary numbers.

ex:

```

<!DOCTYPE html>
<html>
<head>
  <title>MyPage!</title>
</head>
<body>
  <script>
    var a=10, b=15;
    var c = a ^ b;
    document.writeln(c); //5
    /*
      10 - 1010
      15 - 1111
    */
  </script>
</body>
</html>

```

```
-----  
^ - 0101  
*/  
</script>  
</body>  
</html>
```

Bitwise NOT operator (~)

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>MyPage!</title>  
</head>  
<body>  
    <script>  
        var a=~10;  
        document.writeln(a); // -11  
    </script>  
</body>  
</html>
```

ex:

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>MyPage!</title>  
</head>  
<body>  
    <script>  
        var a=~50;  
        document.writeln(a); // -51  
    </script>  
</body>  
</html>
```

ex:

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>MyPage!</title>  
</head>  
<body>  
    <script>  
        var a=~(-10);  
        document.writeln(a); // 9  
    </script>  
</body>  
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var a= 10 >> 2; //10 / 4
        document.writeln(a); //2
    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var a= 100 >> 4; // 100 / 16
        document.writeln(a); //6
    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var a= 100 << 3; // 100 * 8
        document.writeln(a); //800
    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
```

```

        var a= 10 << 6 // 10 * 64
        document.writeln(a); //640
    </script>
</body>
</html>
```

6) Special operators

We have three types of special operators.

operator	description
:?	It is used to make a conditions
new	It is used to create a instance.
typeof	It will return type of a instance.

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        (5>2)?document.writeln("Hi"):document.writeln("Bye");
    </script>
</body>
</html>
```

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        (5>20)?document.writeln("Hi"):document.writeln("Bye");//Bye
    </script>
</body>
</html>
```

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        ((5>20) && (6<10))?document.writeln("Hi"):document.writeln("Bye"); //Bye
    </script>
</body>
</html>
```

```

ex:
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i=10;
        document.writeln(typeof(i)+"<br>"); //number
        var j=true;
        document.writeln(typeof(j)+"<br>"); //boolean
        var k="hi";
        document.writeln(typeof(k)+"<br>"); //String
        var l=null;
        document.writeln(typeof(l)+"<br>"); //object
        var m;
        document.writeln(typeof(m)+"<br>"); //undefined
    </script>
</body>
</html>

```

Q) Difference between null and undefined?

In JavaScript, undefined is a type, whereas null is an object.

undefined

It means a variable declared, but no value has been assigned a value.

Ex: <script>

```

        var x;
        document.writeln(x); //undefined
    </script>

```

null

A null in JavaScript is an assignment value.

Ex: <script>

```

        var x=null;
        document.writeln(x); //null
    </script>

```

Q) Types of functions in JavaScript?

We have three types of functions.

Named Functions

These types of functions contain name at the time of declaration.

Ex: function f1()

```

    {
        document.writeln("hello world");
    }
    f1();

```

Anonymous Functions

These types of functions do not have any name.

They are declared dynamically at runtime.

```
Ex: var f1=function()
{
document.writeln('hello world');
}
f1();
```

Arrow functions

Arrow functions are more secure when compare to other functions.

As per ES6 standards we need to use arrow function.

```
ex: var f1=()=>{
{
    document.writeln("Hello World");
}
```

JavaScript If Else Stmt

We have three forms of JavaScript if else stmt.

- 1) If stmt
- 2) If else stmt
- 3) if else if stmt

1) If stmt

It will evaluate the code only if our condition is true.

syntax: if(condition)

```
{
-
-
- //code to be evaluate
-
}
```

ex:1

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        if(0,1,2,3,4,5)
        {
            document.writeln("Hello World");
        }
    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
```

```
<body>
    <script>
        if(5,4,3,2,1,0)
        {
            document.writeln("Hello World"); // Nothing
        }
    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        if(-1)
        {
            document.writeln("Hello World");
        }
    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        if((5>2) && (6<10))
        {
            document.writeln("Hello World");
        }
    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
```

```

        if(!(5>2))
    {
        document.writeln("Hello World");//nothing
    }
    </script>
</body>
</html>

```

2) If else stmt

It will evaluate the code either our condition is true or false.

syntax: if(condition)

```

    {
        - // code to be evaluate if cond is true
    }
    else
    {
        - // code to be evaluate if cond is false
    }

```

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        if(true)
    {
        document.writeln("It is true");
    }
    else
    {
        document.writeln('It is false');
    }
    </script>
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        if(false)
    {
        document.writeln("It is true");
    }

```

```

        else
        {
            document.writeln('It is false');
        }
    </script>
</body>
</html>

```

Q) Write a javascript program to find out given number is even or odd without using modules operator (%)?

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var val=prompt("Enter the number :");

        var n=parseInt(val); // 10

        if( (n & 1) == 0)
            document.writeln("It is even");
        else
            document.writeln("It is odd");

        /*
        10 - 1010
        1 - 0001
        -----
        & - 0000
        */
    </script>
</body>
</html>

```

Q) Write a javascript program to check given number is positive or negative?

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var val=prompt("Enter the number :");

        var n=parseInt(val); // 10

```

```

        if(n>0)
            document.writeln("It is positive number");
        else
            document.writeln("It is negative number");
    </script>
</body>
</html>

```

iii) if else if statement

It will execute the code based on multiple conditions.

syntax: if(cond1)

```

{
    - //code to be evaluate if cond1 is true
}
else if(cond2)
{
    - //code to be evaluate if cond2 is true
}
else if(cond3)
{
    - //code to be evaluate if cond3 is true
}
else
{
    - //code to be evaluate if all conditions are false
}

```

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var val=prompt("Enter the option :");

        var n=parseInt(val); // 10

        if(n==100)
            document.writeln("It is police number");
        else if(n==103)
            document.writeln("It is enquiry number");
        else if(n==108)
            document.writeln("It is emergency number");
        else
            document.writeln("invalid option");
    </script>
</body>
</html>

```

JavaScript Switch

It is similar to if else if statement.

It will evaluate the code base on multiple conditions.

syntax: switch(condition)

```
{  
    case1 value: //code to be execute  
        break stmt;  
    case2 value: //code to be execute  
        break stmt;  
    -  
    default: //code to be execute  
}
```

ex:

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>MyPage!</title>  
</head>  
<body>  
    <script>  
        var val=prompt("Enter the option :");  
  
        var n=parseInt(val); // 10  
  
        switch(n)  
        {  
            case 100: document.writeln("It is police number"); break;  
            case 103: document.writeln("It is enquiry number"); break;  
            case 108: document.writeln("It is emergency number"); break;  
            default: document.writeln("invalid option");  
        }  
    </script>  
</body>  
</html>
```

ex:

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>MyPage!</title>  
</head>  
<body>  
    <script>  
        var ch=prompt("Enter the character :");  
        switch(ch)  
        {  
            case 'a': document.writeln("It is a vowel"); break;  
            case 'e': document.writeln("It is a vowel"); break;  
            case 'i': document.writeln("It is a vowel"); break;
```

```

        case 'o': document.writeln("It is a vowel"); break;
        case 'u': document.writeln("It is a vowel"); break;
        default: document.writeln("It is not a vowel");
    }
</script>
</body>
</html>

ex:
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var str=prompt("Enter the String :");

        switch(str)
        {
            case "one": document.writeln("It is a January"); break;
            case "two": document.writeln("It is a February"); break;
            case "three": document.writeln("It is a March"); break;
            case "four": document.writeln("It is a April"); break;
            case "five": document.writeln("It is a May"); break;
            default: document.writeln("coming soon... ");
        }
    </script>
</body>
</html>
```

JavaScript LOOPS

We have four types of loops in javascript.

- 1) do while loop
- 2) while loop
- 3) for loop
- 4) for in loop

1) do while loop

It will evaluate the code until our condition is true.

In do while loop, our code will execute atleast for one time.

syntax: do

```

    {
        -
        - //code to be evaluate
        -
    }while(condition);
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i=1;
        do
        {
            document.writeln(i+" ");
            i++;
        }
        while(i<=10);
    </script>
</body>
</html>
```

Q) Write a javascript program to perform sum of 10 natural numbers?

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i=1, sum=0;
        do
        {
            sum+=i;
            i++;
        }
        while(i<=10);
        document.writeln(sum);
    </script>
</body>
</html>
```

2) while loop

It will evaluate the code until our condition is true.

syntax: while(condition)

```
{
    -
    - //code to be evaluate
    -
}
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i=10;
        while(i>=1)
        {
            document.writeln(i+" "); //10 9 8 7 6 5 4 3 2 1
            i--;
        }
    </script>
</body>
</html>
```

Q) Write a javascript program to dispaly multiplication table of a given number?

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var n=5;
        var i=1;
        while(i<=10)
        {
            document.writeln(n+" * "+i+" = "+n*i+"  
");
            i++;
        }
    </script>
</body>
</html>
```

3) for loop

It will evalute the code untill our condition is true.

syntax: for(initialization;condition;incrementation/decrementation)

```
{
    -
    - //code to be evaluate
    -
}
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
```

```

</head>
<body>
    <script>
        for(var i=1;i<=10;i++)
        {
            document.writeln(i+" ");
        }
    </script>
</body>
</html>

```

Q) Write a javascript program to find out factorial of a given number?

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var n=5,fact=1;
        for(var i=n;i>=1;i--)
        {
            fact*=i;
        }
        document.writeln(fact);
    </script>
</body>
</html>

```

4) for in loop

It is used to iterate elements from array.

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var arr=[10,20,30];
        for(var i in arr)
        {
            document.writeln(arr[i]+" ");
        }
    </script>
</body>
</html>

```

```

ex:
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var arr=["hi","hello","bye"]
        for(var i in arr)
        {
            document.writeln(arr[i]+ " ");
        }
    </script>
</body>
</html>

```

Note:

If number of iterations are known by the user then we need to use for loop.

If number of iterations are not known by the user then we need to use while loop.

If number of iterations are not known by the user but code must execute atleast for one time then we need to use do while loop.

Javascript Functions

Function is a block of scope which is used to perform particular task.

Functions are designed to maintain business logic.

Using functions we can achieve reusability.

To declare a function in javascript we will use function keyword.

We can declare a function in javascript followed by a function name, and parenthesis().

Javascript function parenthesis contains list of arguments seperated with comma.

syntax: function <function_name>()

```

{
    -
    - //code to be evaluate
    -
}

```

Functions are executed at the time when they we have requested or when event occur.

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        function f1()
        {
            document.bgColor="red";
        }
    </script>

```

```

//calling
f1();

</script>
</body>
</html>

ex:
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        function f1()
        {
            for(var i=1;i<=4;i++)
            {
                for(var j=1;j<=4;j++)
                {
                    document.writeln("*");
                }
                //new line
                document.writeln("<br>");
            }
        }
        //calling
        f1();
    </script>
</body>
</html>

```

```

ex:
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        function f1()
        {
            for(var i=1;i<=4;i++)
            {
                for(var j=1;j<=i;j++)
                {
                    document.writeln("*");
                }
                //new line

```

```

        document.writeln("<br>");
    }
}
//calling
f1();
f1();

</script>
</body>
</html>
```

In javascript ,every function is a case sensitive.

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        function f1()
        {
            document.writeln("F1 Function");
        }
    //calling
    F1();
    </script>
</body>
</html>
```

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <button onclick="f1()"> Click Here </button>
    <script>
        function f1()
        {
            document.bgColor="blue";
        }
    </script>
</body>
</html>
```

Q) What is JavaScript Closure?

A closure is the combination of a functions bundled together along lexical scope.

In JavaScript, closures are created every time when functions are created.

In other words, a closure gives you access to an outer function's scope from an inner function.

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        //lexical scope
        var a=10;
        function f1()
        {
            //lexical scope
            var b=20;

            function f2()
            {
                var c=30;

                document.writeln(a+" "+b+" "+c);
            }
            //calling
            f2();
        }

        //calling
        f1();
    </script>
</body>
</html>
```

Javascript Object

A javascript object is an entity which is having states and behaviours.

In general, javascript object is a collection of properties and functions.

Javascript is a object based language because everything is present in objects.

Javascript is a template based but not class based.We don't need to create a class to get the object.We can create object directly.

There are three ways to create javascript objects.

- 1) By using Object literal
- 2) By creating instance of an Object i.e using new keyword.
- 3) By using Object constructor i.e using new keyword.

1) By using Object literal

```
<!DOCTYPE html>
<html>
```

```

<head>
    <title>IHUB Talent</title>
</head>
<body>
    <script type="text/javascript">
        emp={
            eid:101,
            ename:"Alan Morries",
            esal:10000
        };
        document.writeln("Employee Id:"+emp.eid+"<br>");
        document.writeln("Employee Name:"+emp.ename+"<br>");
        document.writeln("Employee Salary:"+emp.esal+"<br>");
    </script>
</body>
</html>

```

2) By creating instance of an Object

```

<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            var emp=new Object();
            emp.eid=102;
            emp.ename="Erick Anderson";
            emp.esal=20000;
            document.writeln("Employee Id:"+emp.eid+"<br>");
            document.writeln("Employee Name:"+emp.ename+"<br>");
            document.writeln("Employee Salary:"+emp.esal+"<br>");

        </script>
    </body>
</html>

```

3) By using Object constructor

Here we need to create a function with parameters and each parameter must assign in the current object by using this keyword.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            function emp(eid,ename,esal)
            {

```

```

        this.eid=eid;
        this.ename=ename;
        this.esal=esal;
    }
e=new emp(103,"Ana Julie",30000);
document.writeln("Employee Id :"+e.eid+"  
");
document.writeln("Employee Name :"+e.ename+"  
");
document.writeln("Employee Sal :"+e.esal+"  
");
</script>
</body>
</html>
```

Javascript Array

In javascript , Array is an object which contains similar elements.

Array index always starts with '0' because it is a logical process.

There are three ways to create an array in javascript.

- 1) By using array literal
- 2) By creating instance of an array i.e using new operator.
- 3) By creating array constructor i.e using new operator.

1) By using array literal

```
<!DOCTYPE html>
<html>
<head>
    <title>IHUB Talent</title>
</head>
<body>
    <script type="text/javascript">
        var arr=[10,20,30,40];
        for(var i=0;i<arr.length;i++)
        {
            document.writeln(arr[i]+" ");
        }
    </script>
</body>
</body>
</html>
```

ex:2

```
<!DOCTYPE html>
<html>
<head>
    <title>IHUB Talent</title>
</head>
<body>
    <script type="text/javascript">
        var arr=[10,20,30,40];
        for(var i in arr)
        {
            document.writeln(arr[i]+" ");
```

```

        }
    </script>
</body>
</body>
</html>

ex:3
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            var arr=["html","css","js"];
            for(var i in arr)
            {
                document.writeln(arr[i]+" ");
            }
        </script>
    </body>
    </body>
</html>

```

2) By creating instance of an array i.e using new operator

```

<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            var arr=new Array();
            arr[0]=10;
            arr[1]=20;
            arr[2]=30;
            for(var i in arr)
            {
                document.writeln(arr[i]+" ");
            }
        </script>
    </body>
    </body>
</html>

```

3) By creating array constructor i.e using new operator

```

<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>

```

```
</head>
<body>
    <script type="text/javascript">
        var arr=new Array(10,20,30,40,50);
        for(var i in arr)
        {
            document.writeln(arr[i]+" ");
        }
    </script>
</body>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>
</head>
<body>
    <script type="text/javascript">
        var arr=[];
        arr.push(10);
        arr.push(20);
        arr.push(30);
        for (i in arr)
        {
            document.write(arr[i]+" ");
        }
    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>
</head>
<body>
    <script type="text/javascript">
        var arr=[];
        arr.push(10);
        arr.push(20);
        arr.push(30);
        arr.pop();
        for (i in arr)
        {
            document.write(arr[i]+" ");
        }
    </script>
</body>
</html>
```

```
</script>
</body>
</html>
```

Javascript String

In javascript , string is an object which contains collection of characters.

There are two ways to create a string in javascript.

- 1) By using string literal
- 2) By creating instance of a string.

1) By using string literal

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            var str1="bhaskar";
            document.writeln(str1+"<br>");
            var str2='solution';
            document.writeln(str2+"<br>");
        </script>
    </body>
</body>
</html>
```

2) By creating instance of a string.

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            var str=new String("bhaskar");
            document.writeln(str);
        </script>
    </body>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
```

```
<script type="text/javascript">
    var str="bhaskar";
    document.writeln(str.length);
</script>
</body>
</body>
</html>
```

ex:2

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            var str="bhaskar";
            document.writeln(str.toUpperCase());
            var str2="BHASKAR";
            document.writeln(str.toLowerCase());
        </script>
    </body>
    </body>
</html>
```

ex:3

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            var str1="ihub";
            var str2="talent";
            document.writeln(str1.concat(str2));
        </script>
    </body>
    </body>
</html>
```

ex:4

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            var str1="ihub";
```

```
        document.writeln(str1.charAt(2));
    </script>
</body>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            var str="ihub";
            var arr=str.split("");
            for(var i in arr)
            {
                document.writeln(arr[i]+"<br>");
            }
        </script>
    </body>
    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            var str="ihub";
            var arr=str.split("");
            for(var i=arr.length-1;i>=0;i--)
            {
                document.writeln(arr[i]);
            }
        </script>
    </body>
    </body>
</html>
```

Q) Write a javascript program to perform sum of array elements?

```
<!DOCTYPE html>
<html>
<head>
    <title>My Page!</title>
</head>
```

```

<body>
    <script>
        var arr=[10,20,30,40];
        var sum=0;
        for(var i in arr)
        {
            sum+=arr[i];
        }
        document.writeln(sum);
    </script>
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var str="racar";
        var carr=str.split("// h e l l o");
        var rev="";
        //reading reverse
        for(var i=carr.length-1;i>=0;i--)
        {
            rev+=carr[i];
        }
        if(str==rev)
            document.writeln("It is palindrome ");
        else
            document.writeln("it is not palindrome");
    </script>
</body>
</html>

```

BOM (Browser Object Model)

The Browser Object Model is used to interact with browser.

The default object for a browser is window object. It means we can call all the functions by using window or directly.

ex: window.alert("Welcome to JavaScript");
 or
 alert("Welcome to JavaScript");

window object

It is used to create a window on a browser.

A window object is created automatically by the browser.

A "window" is a object of browser but not javascript.

Javascript objects are String, Array, Date and etc.

A "window" object is used to write programming related to browser.

With the help of window object we can perform following activities very easily.

- 1)It display dialog boxes and pop boxes.
- 2)We can find width and height of a browser.
- 3)We can move or resize the browser.
- 4)Scroll to the browser.
- 5)Get URL,hostname,protocol and etc of a browser.
- 6)We can get javascript history.

1) alert()

It will display alert dialog box. It has message with ok button.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            function f1()
            {
                alert("Welcome to JavaScript");
            }
        </script>
        <button onclick="f1()">click</button>
    </body>
</html>
```

2) confirm()

It will dispaly confirm dialog box. It has message with ok button and cancel button.

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            function f1()
            {
                var v=confirm("Do you wants to delete ?");
                if(v==true)
                {
                    alert("ok");
                }
                else
                {
                    alert("cancel");
                }
            }
        </script>
    </body>
</html>
```

```

        }
    }
</script>
<button onclick="f1()">delete</button>
</body>
</html>
```

3) **prompt()**

It will display prompt dialog box. It contains message with textfield.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            function f1()
            {
                var v=prompt("Who are you?");
                alert("Welcome :" +v);
            }
        </script>

        <button onclick="f1()">click</button>
    </body>
</html>
```

innerWidth and innerHeight

```

<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            var w=window.innerWidth;
            var h=window.innerHeight;
            document.writeln("Width :" +w + "<br>");
            document.writeln("Height :" +h + "<br>");
        </script>
    </body>
</html>
```

Note: Press "CTRL +" for zoomin.

Press "CTRL + -" for zoomout.

4) window.open()

ex:1

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            function openWindow()
            {
                window.open("http://www.google.com");
            }
        </script>
        <button onclick="openWindow()">open a new window</button>
    </body>
</html>
```

ex:2

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            function openWindow()
            {
                window.open("http://www.google.com","_blank");
            }
        </script>
        <button onclick="openWindow()">open a new window</button>
    </body>
</html>
```

ex:3

```
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            function openWindow()
            {
                window.open("http://www.google.com","_blank","width=200px,height=200px");
            }
        </script>
```

```

        <button onclick="openWindow()">open a new window</button>
    </body>
</html>

close()
<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            var myWindow;
            function openWindow()
            {
                myWindow=window.open("http://www.google.com","","width=300px,height=300px");
            }
            function closeWindow()
            {
                myWindow.close();
            }
        </script>
        <button onclick="openWindow()">open a new window</button>
        <button onclick="closeWindow()">close a window</button>
    </body>
</html>

```

Whenever we open a new window , it takes left top alignment.
In order to move the window we need to use moveTo() or moveBy() function.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            var myWindow;
            function openWindow()
            {
                myWindow=window.open("http://www.google.com","","width=300px,height=300px");
            }
            function moveWindow()
            {
                myWindow.moveTo(100,100);
            }
        </script>

```

```

        </script>

        <button onclick="openWindow()">open a new window</button>
        <button onclick="moveWindow()">move window</button>
    </body>
</html>

```

Note: Here we can't move window because in browser console we will get one error.

To over come this limitation we need to use custom window.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            var myWindow;
            function openWindow()
            {
                myWindow=window.open("", "_blank", "width=300px,height=300px");

                }
            function moveWindow()
            {
                myWindow.moveTo(100,100);
            }

        </script>

        <button onclick="openWindow()">open a new window</button>
        <button onclick="moveWindow()">move window</button>
    </body>
</html>

```

Note: MoveTo() function will move from absolute position.
MoveBy() function will move from relative position.

setTimeout()

The setTimeout() is executed only once.

If you need repeated executions, use setInterval() instead.

ex:

```
<!DOCTYPE html>
```

```

<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            function f1()
            {
                setTimeout(function f1()
                {
                    alert("Hello World")
                },4000);
            }
        </script>

        <button onclick="f1()">click</button>

    </body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
    </head>
    <body>
        <script type="text/javascript">
            function setTimeOut()
            {
                setTimeout(Anim,4000);
            }
            function Anim()
            {
                alert("Yahoo! this is javascript");
            }
        </script>

        <button onclick="setTimeOut()">click</button>

    </body>
</html>

```

`clearTimeout()`

The `clearTimeout()` method clears a timer set with the `setTimeout()` method.

ex:

```

<!DOCTYPE html>
<html>

```

```

<head>
    <title>IHUB Talent</title>
</head>
<body>
    <script type="text/javascript">
        var myId;
        function setTimeOut()
        {
            myId=setTimeout(Anim,4000);
        }
        function Anim()
        {
            alert("Yahoo! this is javascript");
        }
        function removeTimeOut()
        {
            clearTimeout(myId);
        }
    </script>
    <button onclick="setTimeOut()">set time</button>
    <button onclick="removeTimeOut()">remove time</button>
</body>
</html>

```

setInterval()

A setInterval() method calls a function to evaluate the expression at specified interval(milliseconds).

A setInterval() method calls continuously function untill we call clearInterval() method or window is closed.

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
        <style>
            div
            {
                width:150px;
                height: 150px;
                background-color: #FF0000;
            }
        </style>
    </head>
    <body>
        <script type="text/javascript">
            var a=0;
            setInterval(Anim,1000);
        </script>
    </body>

```

```

        function Anim()
        {
            a = a + 10;
        var target=document.getElementById("myId");
            target.style.marginLeft= a + 'px';
        }

</script>

<div id="myId"></div>
</body>
</html>

```

clearInterval()

A clearInterval() function is used to clear the timer set on setInterval() function.
An id which is return from setInterval() function will use as parameter to clearInterval().

ex:

```

<!DOCTYPE html>
<html>
    <head>
        <title>IHUB Talent</title>
        <style>
            div
            {
                width:150px;
                height: 150px;
                background-color: #FF0000;
            }
        </style>
    </head>
    <body>
        <script type="text/javascript">
            var a=0;
            var id=setInterval(Anim,1000);

            function Anim()
            {
                a = a + 10;
                if(a==100)
                {
                    clearInterval(id);
                }

                var target=document.getElementById("myId");
                target.style.marginLeft= a + 'px';
            }

        </script>

        <div id="myId"></div>
    </body>
</html>

```

```

        </body>
    </html>

window history
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
    <style>
        body
        {
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
        }
        a
        {
            text-decoration: none;
            color:blue;
        }
    </style>
</head>
<body>
    &laquo;
    <a href="javascript:history.back()">Previous</a>
    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
    <a href="javascript:history.forward()">Next</a>
    &raquo;
</body>
</html>

```

localStorage

A localStorage properties allows us to save key/value pairs in a browser window.

A localStorage allows us to store the data with no-expiry. It means our data will not be deleted even if we close the browser. It will be present for next day.

A localStorage is a read-only.

To add the data in a localStorage we need to use `setItem(key,value)` function.

To read the data from localStorage we need to use `getItem(key)` function.

To remove particular data from localStorage we need to use `removeItem(key)` function.

To remove all the data from localStorage we need to use `clear()` function.

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        localStorage.setItem("name", "Alan");
    </script>

```

```

localStorage.setItem("age",25);
document.writeln(localStorage.getItem("name")+"<br>");
document.writeln(localStorage.getItem("age")+"<br>");

localStorage.removeItem("age");
document.writeln(localStorage.getItem("name")+"<br>");
document.writeln(localStorage.getItem("age")+"<br>");

localStorage.clear();
document.writeln(localStorage.getItem("name")+"<br>");
document.writeln(localStorage.getItem("age")+"<br>");

</script>
</body>
</html>

```

sessionStorage

A sessionStorage properties allows us to save key/value pair in a browser window.

A sessionStorage store the data with respect to one session. It means our data will be deleted once if we close the browser window.

To add the data in a sessionStorage we need to use `setItem(key,value)` function.

To read the data from sessionStorage we need to use `getItem(key)` function.

To remove particular data from sessionStorage we need to use `removeItem(key)` function.

To remove all the data from sessionStorage we need to use `clear()` function.

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        sessionStorage.setItem("name", "Jose");
        sessionStorage.setItem("age",27);
        document.writeln(sessionStorage.getItem("name")+"<br>");
        document.writeln(sessionStorage.getItem("age")+"<br>");

        sessionStorage.removeItem("age");
        document.writeln(sessionStorage.getItem("name")+"<br>");
        document.writeln(sessionStorage.getItem("age")+"<br>");

        sessionStorage.clear();
        document.writeln(sessionStorage.getItem("name")+"<br>");
        document.writeln(sessionStorage.getItem("age")+"<br>");

    </script>
</body>
</html>

```

DOM

The document object represent whole HTML document.

When HTML document is loaded in a browser it represent document object.

Here HTML document is represented in a tree node hierarchy.

A document object is a root node for entire html document.

DOM always looks for three nodes.

- 1)Element node
- 2)Attribute node
- 3)Text node

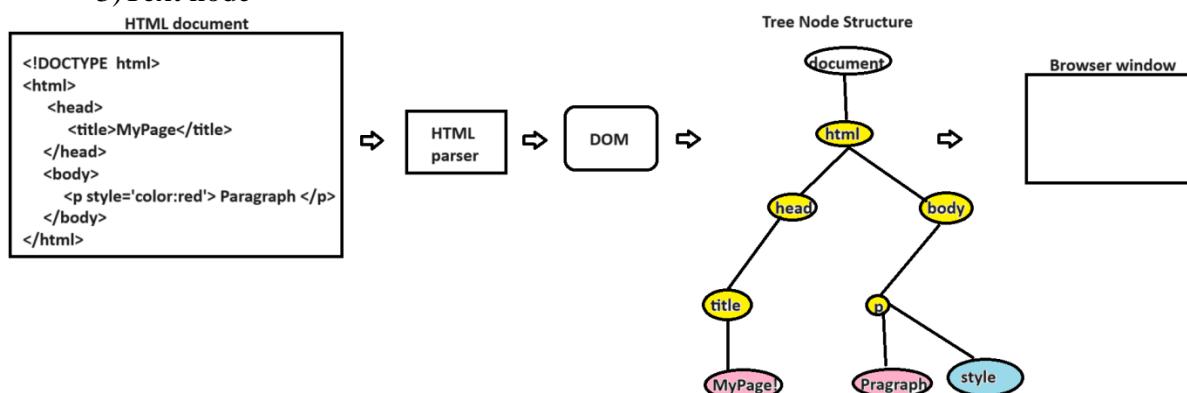


Diagram: class22.1

Using document object we can add dynamic content to the web page.

A document object is a property of window. It means we can call document object directory or by using window.

ex: window.document
 or
 document

document.write()

It is used to display data or custom messages without space.

ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
  </head>
  <body>
    <script type="text/javascript">
      document.write("This is First Stmt");
      document.write("This is Second Stmt");
    </script>
  </body>
</html>
```

document.writeln()

It will display the output with space at last.

ex:

```
<!DOCTYPE html>
<html>
  <head>
```

```

        <title>MyPage!</title>
    </head>
    <body>
        <script type="text/javascript">
            document.writeln("This is First Stmt");
            document.writeln("This is Second Stmt");
        </script>
    </body>
</html>

```

document.getElementById()

It is used to read the elements based on id.

ex:1

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    Name: <input type="text" id="t1"/>
    <button onclick="f1()"> submit </button>
    <script>
        function f1()
        {
            var val=document.getElementById('t1').value;
            document.writeln("Welcome :" + val);
        }
    </script>
</body>
</html>

```

ex:2

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    Name: <input type="text" id="t1"/>
    <button onclick="f1()"> submit </button>
    <br>
    <h1 id="result"></h1>
    <script>
        function f1()
        {
            var val=document.getElementById('t1').value;
            document.getElementById('result').innerHTML="Welcome
            :" + val;
        }
    </script>
</body>
</html>

```

```

        </script>
</body>
</html>

Adding two text fields and display the result using javascript
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>

    <table align="center">
        <tr>
            <td>No1:</td>
            <td><input type="text" id="t1"/></td>
        </tr>
        <tr>
            <td>No2:</td>
            <td><input type="text" id="t2"/></td>
        </tr>
        <tr>
            <td>&nbsp;</td>
            <td><button onclick="f1()">ADD</button></td>
        </tr>
        <tr>
            <td>Result:</td>
            <td><h2 id="result"></h2></td>
        </tr>
    </table>
    <script type="text/javascript">

        function f1()
        {
            var val1=document.getElementById('t1').value;
            var val2=document.getElementById('t2').value;
            var a=parseInt(val1)
            var b=parseInt(val2);
            var c=a+b;
            document.getElementById('result').innerHTML=c;
        }
    </script>
</body>
</html>

```

Hide and show the portion of a form page using javascript

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>

```

```

</head>
<body>

    <fieldset id="cur_id">
        <legend>Current Address:</legend>
        <table align="center">
            <tr>
                <td>House No:</td>
                <td><input type="text" id="t1"/></td>
            </tr>
            <tr>
                <td>Locality:</td>
                <td><input type="text" id="t2"/></td>
            </tr>
            <tr>
                <td>State:</td>
                <td><input type="text" id="t3"/></td>
            </tr>
        </table>
    </fieldset>
    <br>

    <input type="checkbox" id="box" onclick="f1()" />
    Permanent Address same as Current Address?

    <br>
    <br>

    <fieldset id="per_id">
        <legend>Permanent Address:</legend>
        <table align="center">
            <tr>
                <td>House No:</td>
                <td><input type="text" id="t1"/></td>
            </tr>
            <tr>
                <td>Locality:</td>
                <td><input type="text" id="t2"/></td>
            </tr>
            <tr>
                <td>State:</td>
                <td><input type="text" id="t3"/></td>
            </tr>
        </table>
    </fieldset>

    <script type="text/javascript">
        function f1()
        {
            if(document.getElementById('box').checked)
            {

```

```

        document.getElementById('per_id').style.display="none";
    }
    else
    {
        document.getElementById('per_id').style.display="block";
    }
</script>

</body>
</html>

```

Ex:-

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <fieldset id="cur_id">
        <legend>Current Address:</legend>
        <table align="center">
            <tr>
                <td>House No:</td>
                <td><input type="text" id="t1"/></td>
            </tr>
            <tr>
                <td>Locality:</td>
                <td><input type="text" id="t2"/></td>
            </tr>
            <tr>
                <td>State:</td>
                <td><input type="text" id="t3"/></td>
            </tr>
        </table>
    </fieldset>
    <br>

    <input type="checkbox" id="box" onclick="f1()" />
    Permanent Address same as Current Address?

    <br>
    <br>

    <fieldset id="per_id">
        <legend>Permanent Address:</legend>
        <table align="center">
            <tr>

```

```

        <td>House No:</td>
        <td><input type="text" id="t4"/></td>
    </tr>
    <tr>
        <td>Locality:</td>
        <td><input type="text" id="t5"/></td>
    </tr>
    <tr>
        <td>State:</td>
        <td><input type="text" id="t6"/></td>
    </tr>
</table>
</fieldset>

<script type="text/javascript">
    function f1()
    {
        if(document.getElementById('box').checked)
        {
            var val1=document.getElementById('t1').value;
            var val2=document.getElementById('t2').value;
            var val3=document.getElementById('t3').value;

            document.getElementById('t4').value=val1;
            document.getElementById('t5').value=val2;
            document.getElementById('t6').value=val3;
        }
        else
        {
            document.getElementById('t4').value="";
            document.getElementById('t5').value="";
            document.getElementById('t6').value="";
        }
    }
</script>

</body>
</html>

```

document.getElementsByName()

The getElementsByName() method returns a collection of elements with a specified name.

ex:1

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    Name: <input type="text" name="t1"/>

```

```

<button onclick="f1()"> click </button>
<br>
<h1 id="result"></h1>

<script>
    function f1()
    {
        var name=document.getElementsByName('t1')[0].value;
        document.getElementById('result').innerHTML=name;
    }
</script>
</body>
</html>

```

ex:2

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    Name: <input type="text" name="t1"/>
    <button onclick="f1()"> click </button>
    <br>
    <h1 id="result"></h1>
    <script>
        function f1()
        {
            var name=document.getElementsByName('t1')[0].tagName;
            document.getElementById('result').innerHTML=name;
        }
    </script>
</body>
</html>

```

ex:3

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    First Name: <input type="text" name="t1"/>
    Last Name: <input type="text" name="t1"/>
    <button onclick="f1()"> click </button>
    <br>
    <h1 id="result"></h1>

    <script>
        function f1()

```

```

        {
            var name=document.getElementsByName('t1').length;
            document.getElementById('result').innerHTML=name;
        }
    </script>
</body>
</html>

```

ex:4

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <h2>Course Completed?</h2>
    <button onclick="f1()">Select All</button>
    <br>
    <input type="checkbox" name="box" value="html"/> HTML <br>
    <input type="checkbox" name="box" value="css"/> CSS <br>
    <input type="checkbox" name="box" value="js"/> JavaScript <br>
    <input type="checkbox" name="box" value="bs"/> Bootstrap <br>
    <script>
        function f1()
        {
            var x=document.getElementsByName('box');
            for(var i=0;i<x.length;i++)
            {
                if(x[i].type=="checkbox")
                {
                    x[i].checked=true;
                }
            }
        }
    </script>
</body>
</html>

```

document.getElementsByTagName()

The `getElementsByTagName()` method returns a collection of all elements with a specified tag name.

ex:1

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    Name : <input type="text" name="t1" id="t1"/>
    <button onclick="f1()"> click </button>

```

```

<h1 id="result"></h1>
<script>
    function f1()
    {
        var name=document.getElementsByTagName('input')[0].value;
        document.getElementById('result').innerHTML=name;
    }
</script>
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <div> This is div1 </div>
    <div> This is div2 </div>
    <div> This is div3 </div>
    <button onclick="f1()"> change </button>
    <script>
        function f1()
        {
            var x=document.getElementsByTagName('div');
            x[0].innerHTML="This is change1";
            x[1].innerHTML="This is change2";
            x[2].innerHTML="This is change3";
        }
    </script>
</body>
</html>

```

ex:3

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <div> This is div1 </div>
    <div> This is div2 </div>
    <div> This is div3 </div>
    <button onclick="f1()"> change </button>
    <script>
        function f1()
        {
            var x=document.getElementsByTagName('div');
            x[0].style.textAlign="center";
        }
    </script>
</body>
</html>

```

```

        x[1].style.color="blue";
        x[2].style.backgroundColor="yellow";
    }
</script>
</body>
</html>

```

document.getElementsByClassName()

The `getElementsByClassName()` method returns a collection of child elements with a given class name.

ex:1

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
    <style>
        .c1
        {
            text-align: center;
            color:blue;
            background-color: yellow;
        }
    </style>
</head>
<body>
    <div class="c1"> This is Division Tag </div>
    <button onclick="f1()"> click </button>
    <h1 id="result"></h1>
    <script>
        function f1()
        {
            var name=document.getElementsByClassName('c1').length;
            document.getElementById('result').innerHTML=name;
        }
    </script>
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
    <style>
        .c1
        {
            text-align: center;
            color:blue;
            background-color: yellow;
        }
    </style>

```

```

        </style>
</head>
<body>
    <div class="c1"> This is Division Tag </div>
    <button onclick="f1()"> click </button>
    <h1 id="result"></h1>
    <script>
        function f1()
        {
            var x=document.getElementsByClassName('c1');
            x[0].innerHTML="This is css";
            x[0].style.backgroundColor="cyan";
        }
    </script>
</body>
</html>

```

addEventListener()

It is used to add the handler to the function.

ex:1

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <h1> Click Here </h1>
    <script>

        addEventListener("click",function(){
            alert("You Have Clicked");
        });
    </script>
</body>
</html>

```

ex:2

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <h1> Click Here </h1>
    <script>
        addEventListener("click",f1);
        function f1()
        {
            alert("Yahoo! You have clicked");
        }
    </script>

```

```

        </script>
</body>
</html>

ex:3
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <h1 id="hover"> Mouse Over Here </h1>
    <br>
    <p id="result"></p>
    <script>
        var y=document.getElementById('hover');
        y.addEventListener('mouseover',f1);
        function f1()
        {
            document.getElementById('result').innerHTML="Mouse over here";
        }
    </script>
</body>
</html>

ex:4
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <h1 id="hover"> Mouse Over Here </h1>
    <br>
    <p id="result"></p>
    <script>
        var y=document.getElementById('hover');
        y.addEventListener('mouseout',f1);
        function f1()
        {
            document.getElementById('result').innerHTML="Mouse out here";
        }
    </script>
</body>
</html>
```

removeEventListener()

It is used to remove the handler from function.

ex:

```
<!DOCTYPE html>
```

```

<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <h1 id="hover"> Mouse Over Here </h1>
    <button onclick="f2()"> stop </button>
    <br>
    <p id="result"></p>
    <script>
        var y=document.getElementById('hover');
        y.addEventListener('mouseover',f1);
        function f1()
        {
            document.getElementById('result').innerHTML+="<div>Mouse over here</div>";
        }
        function f2()
        {
            y.removeEventListener('mouseover',f1);
            document.getElementById('result').innerHTML+="Event Stopped!";
        }
    </script>
</body>
</html>

```

Javascript program to convert feet to inches using addEventListener

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <table align="center">
        <tr>
            <td>Feet</td>
            <td>&ampnbsp</td>
            <td>Inches</td>
        </tr>
        <tr>
            <td><input type="text" id="feet"/></td>
            <td><big>=</big></td>
            <td><input type="text" id="inches"/></td>
        </tr>
    </table>
    <script>
        var feet=document.getElementById("feet");
        var inches=document.getElementById("inches");

```

```

feet.addEventListener('input',function(){
    var f=this.value;
    var i=f*12;
    inches.value=i;
});

inches.addEventListener('input',function(){
    var i=this.value;
    var f=i/12;
    if(!Number.isInteger(f))
    {
        f = f.toFixed(2);
    }
    feet.value=f;
})

```

</script>

</body>
</html>

JavaScript Date

A javascript Date object is used to display Date and time.
Using javascript javascript Date object we can display timer also.

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var date=new Date();
        document.writeln(date);
    </script>
</body>
</html>

```

ex:2

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var d=new Date();
        var dd=d.getDate();
        var mm=d.getMonth()+1;
        var yyyy=d.getFullYear();
    </script>

```

```
        document.writeln(dd+"/"+mm+"/"+yyyy);
    </script>
</body>
</html>
```

ex:3

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var d=new Date();
        var h=d.getHours();
        var m=d.getMinutes();
        var s=d.getSeconds();
        document.writeln(h+":"+m+":"+s);
    </script>
</body>
</html>
```

ex:4

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
    <style>
        body
        {
            height:100vh;
            background: linear-gradient(yellow,red);
            display:flex;
            justify-content: center;
            align-items: center;
        }
        span
        {
            box-shadow:2px 2px 12px 6px #FFF;
            width:147px;
            height:42px;
            font-size:40px;
            color:#FFF;
            font-weight:bold;
        }
    </style>
</head>
<body>
    <span id="result"></span>
```

```

<script>
    window.onload= function(){ getTime(); }
    function getTime()
    {
        var d=new Date();
        var h=d.getHours();
        var m=d.getMinutes();
        var s=d.getSeconds();
        m=check(m);
        s=check(s);

document.getElementById('result').innerHTML=h+":"+m+":"+s;
        setInterval(getTime,1000);
    }

    function check(i)
    {
        if(i<10)
        {
            i = "0"+i;
        }
        return i;
    }
</script>
</body>
</html>

```

Interview Questions

Q) What is the difference between innerText vs innerHTML ?

innerText

The innerText property is used to write the simple text using JavaScript dynamically.

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <button onclick="f1()"> click Here </button>
    <br>
    <div id="result"></div>
    <script>
        function f1()
        {
            document.getElementById('result').innerText=<p>This is javascript
innerText</p>";
        }
    </script>
</body>
</html>

```

innerHTML:

The innerHTML property is used to write the HTML code using JavaScript dynamically.

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <button onclick="f1()"> click Here </button>
    <br>
    <div id="result"></div>
    <script>
        function f1()
        {
            document.getElementById('result').innerHTML=<p
style='color:red'>This is javascript innerHTML</p>;
        }
    </script>
</body>
</html>
```

Javascript program to hide and show the password in a textfield

```
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- fontawesome icon cdn link -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css" />

    <style type="text/css">
        .myClass
        {
            padding-right:25px;
        }
        #myId
        {
            position: relative;
            right:25px;
        }
    </style>
</head>
<body>
    Password: <input type="password" id="t1" class="myClass"/>
    <span class="fa fa-eye" id="myId" onclick="f1()"></span>
    <script>
        function f1()
        {
```

```

        var x=document.getElementById('t1');
        if(x.type=="password")
        {
            x.type="text";
        }
        else
        {
            x.type="password";
        }
    }
</script>
</body>
</html>
```

JavaScript Regular Expression

Regular expressions are patterns used to match character combinations in strings.
In JavaScript, regular expressions are also objects.

JavaScript Form validation using RegularExpression

To generate proper regular expression we can login below url.

ex: https://regex101.com/

ex:

```
<!DOCTYPE html>
<html>
<style>
input[type=text],input[type=password], select {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
}
```

```
input[type=submit] {
  width: 100%;
  background-color: #4CAF50;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}
```

```
input[type=submit]:hover {
  background-color: #45a049;
}
```

```
div {
```

```

border-radius: 5px;
background-color: #f2f2f2;
padding: 20px;
width:500px;
position: relative;
left:200px;
top:20px;
}
</style>

<script type="text/javascript">

function validate()
{
    var name=document.getElementById('name').value;
    var pwd=document.getElementById('pwd').value;
    var phone=document.getElementById('phone').value;
    var email=document.getElementById('email').value;
    var country=document.getElementById('country').value;

    var namecheck=/[A-Za-z. ]{6,20}$/;
    var pwdcheck=/([A-Z][a-zA-Z0-9!@#$%^&*]{10,30})$/;
    var phonecheck=/[789][0-9]{9}$/;
    var emailcheck=/[A-Za-z.]{1,}@[A-Za-z]{2,15}[.][A-Za-z]{3,}/;

    if(!(namecheck.test(name)))
    {
        alert("UserName must be 6 characters");
        document.getElementById('name').value="";
        document.getElementById('name').focus();
        return false;
    }

    if(!(pwdcheck.test(pwd)))
    {
        alert("password must have 1 uppercase, 1 special symbol and 1 digit");
        document.getElementById('pwd').value="";
        document.getElementById('pwd').focus();
        return false;
    }

    if(!(phonecheck.test(phone)))
    {
        alert("Phone must start with 7,8,9 series with 10 digits");
        document.getElementById('phone').value="";
    }
}

```

```
document.getElementById('phone').focus();
    return false;
}

if(!(emailcheck.test(email)))
{
    alert("Please insert valid email");
    document.getElementById('email').value="";
    document.getElementById('email').focus();
    return false;
}

if(country=="")
{
    alert("Please select the country option ");
    return false;
}
return true;
}

</script>

<body>

<div>
<form action="/action_page.php" onsubmit="validate()">

    <label for="name">UserName</label>
    <input type="text" id="name" name="name" placeholder="Your username.."/>

    <label for="pwd">Password</label>
    <input type="text" id="pwd" name="pwd" placeholder="Your password.."/>

    <label for="phone">Phone</label>
    <input type="text" id="phone" name="phone" placeholder="Your phone.."/>

    <label for="email">Email</label>
    <input type="text" id="email" name="email" placeholder="Your email.."/>

    <label for="country">Country</label>
    <select id="country" name="country">
        <option value="">none</option>
        <option value="australia">Australia</option>
        <option value="canada">Canada</option>
        <option value="usa">USA</option>
    </select>

    <input type="submit" value="Submit">
</form>
```

```
</div>
```

```
</body>  
</html>
```

JavaScript Interview Questions

Synchronous and Asynchronous in JavaScript

Synchronous JavaScript:

As the name suggests synchronous means to be in a sequence, i.e. every statement of the code gets executed one by one. So, basically a statement has to wait for the earlier statement to get executed.

ex:

```
<script>  
    document.write("Hi"); // First  
    document.write("<br>");  
    document.write("IHUB TALENT"); // Second  
    document.write("<br>");  
    document.write("How are you"); // Third  
</script>
```

Asynchronous JavaScript:

Asynchronous code allows the program to be executed immediately where the synchronous code will block further execution of the remaining code until it finishes the current one. This may not look like a big problem but when you see it in a bigger picture you realize that it may lead to delaying the User Interface.

ex:

```
<script>  
    document.write("Hi");  
    document.write("<br>");  
    setTimeout(function() {  
        document.write("Let us see what happens");  
    }, 2000);  
    document.write("<br>");  
    document.write("End");  
    document.write("<br>");  
</script>
```

Javascript promises

Promises are used to handle asynchronous operations in JavaScript.

They can handle multiple asynchronous operations easily and provide better error handling than callbacks and events.

A Promise has four states:

1. fulfilled: Action related to the promise succeeded
2. rejected: Action related to the promise failed
3. pending: Promise is still pending i.e. not fulfilled or rejected yet
4. settled: Promise has fulfilled or rejected

A promise can be created using Promise constructor.

Syntax: var promise = new Promise(function(resolve, reject){
 //do something
 });

```

ex:1
<script>
var promise = new Promise(function(resolve, reject) {
    resolve('IHub Talent');
})
promise
    .then(function(successMessage) {
        //success handler function is invoked
        console.log(successMessage);
    }, function(errorMessage) {
        console.log(errorMessage);
    })
</script>

```

```

ex:2
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <script>
            var promise= new Promise(function(resolve,reject){
                reject("Reject");
            })
            promise
            .then(function(successMessage){
                document.writeln(successMessage);

            },function(errorMessage)
            {
                document.writeln(errorMessage);
            })
        </script>
    </body>
</html>

```

Q) What is the difference between var , let and const ?

var

It is a functional scope.
We can declare without initialization.

It can be updated.
It can be redeclared.

It can be accessible without declaration and default value is undefined.

let

It is a block scope.
We can declare without initialization.

It can be updated.
It can't be redeclared.

It can't be accessible without declaration.

const

It is a block scope.
We can't declare without initialization.

It can't be updated.
It can't be redeclared.

It can't be accessible without declaration.

initialization

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var i;
        document.writeln(i); //undefined
    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        let i;
        document.writeln(i); //undefined
    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        const i;
        document.writeln(i); //invalid
    </script>
</body>
</html>
```

update

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
```

```
<body>
  <script>
    var i=10;
    i=20;
    document.writeln(i); //20
  </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
  <title>MyPage!</title>
</head>
<body>
  <script>
    let i=10;
    i=20;
    document.writeln(i); //20
  </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
  <title>MyPage!</title>
</head>
<body>
  <script>
    const i=10;
    i=20;
    document.writeln(i); //invalid
  </script>
</body>
</html>
```

Redeclared

ex:

```
<!DOCTYPE html>
<html>
<head>
  <title>MyPage!</title>
</head>
<body>
  <script>
    var i=10;
    var i=20;
```

```
    document.writeln(i); //20
  </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
  <title>MyPage!</title>
</head>
<body>
  <script>
    let i=10;
    let i=20;
    document.writeln(i); //invalid
  </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
  <title>MyPage!</title>
</head>
<body>
  <script>
    const i=10;
    const i=20;
    document.writeln(i); //invalid
  </script>
</body>
</html>
```

Accessible

ex:

```
<!DOCTYPE html>
<html>
<head>
  <title>MyPage!</title>
</head>
<body>
  <script>
    document.writeln(i); //undefined
    var i=10;
  </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
<title>MyPage!</title>
</head>
<body>
<script>
    document.writeln(i); //invalid!
    let i=10;
</script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        document.writeln(i); //invalid
        const i=10;
    </script>
</body>
</html>
```

Q) What is javascript hoisting?

Hoisting is the default behavior of JavaScript where all the variable and function declarations are moved on top.

This means that irrespective of where the variables and functions are declared, they are moved on top of the scope.

ex:1 i=10;
 document.writeln(i); ==> i=10;
 var i; document.writeln(i);

```
ex:2    f1();                                function f1()
                                                {
                                                document.writeln("Hello");
function f1()
{
    document.writeln("Hello");    ==>
}
}                                              }
```

Q) What is spread operator in JavaScript?

The spread operator is used to spreading an array.

```
<!DOCTYPE html>
<html>
<head>
```

```

<title>MyPage!</title>
</head>
<body>
<script>
    var sum=0;
    function f1(num1,num2,num3,num4)
    {
        sum=sum+num1+num2+num3+num4;
        document.writeln(sum);
    }
    let arr=[10,20,30,40];
    f1(...arr);
</script>
</body>
</html>

```

Q) What is javascript recursion?

A function which call itself for many number of times is called recursion.

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
<script>
    function f1(i)
    {
        if(i<=10)
        {
            document.writeln(i);// 1 2 3 4 5 6 7 8 9 10
            f1(i+1);
        }
    }
    f1(1);
</script>
</body>
</html>

```

Q) What is map() method in javascript?

A map() creates a new array from calling a function for every array element.

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
<script>
    var arr=[10,20,30];
    var newArr=arr.map(f1);
    function f1(ele)

```

```

        {
            return ele;
        }
        document.writeln(newArr);
    </script>
</body>
</html>

ex:
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <script>
            var arr=[10,20,30];
            var newArr=arr.map(f1);
            function f1(ele)
            {
                return ele+100;
            }
            document.writeln(newArr);
        </script>
    </body>
</html>

```

Q) What is reduce function in javascript?

The reduce() method executes a reducer function for array element.

The reduce() method returns a single value.

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        var arr=[100,20,30];
        var res=arr.reduce(f1);
        function f1(total,ele)
        {
            return total-ele;
        }
        document.writeln(res);
    </script>
</body>
</html>

```

Javascript OOPS

OOPS stands for Object oriented Programming System/Structure.

A technology is said to be object oriented if it support following features.

- class
- object
- abstraction
- encapsulation
- inheritance
- and
- polymorphism.

Javascript is not a object oriented programming language. It is a object based programming language because javascript contains objects like Arrays, Strings, RegEx, Date and etc.

class

It is a template for an object.

In javascript, class will not consider as an object.

To declare a class we need to use class keyword following by any particular name.

As per naming conventions in javascript , A class name must starts with uppercase letter.

ex: class Example

```
{  
    -  
    -  
    -  
}
```

constructor

A constructor is a special function which is used to initialized or create an object.

A constructor will be called when memory is allocated.

To declare a constructor we will use constructor keyword.

A class can have only constructor.

ex: class Example

```
{  
    constructor()  
    {  
        -  
        -  
        -  
    }  
}
```

object

Object is an entity which is having state and behaviours (properties and functions).

To create object in javascript we will use new keyword.

syntax: var ref_var=new Object_name();

ex: var e = new Example();

```
<!DOCTYPE html>  
<html>  
    <head>  
        <title>MyPage!</title>
```

```

</head>
<body>
    <script>
        class Example
        {
            constructor()
            {
                document.writeln("constructor <br>");
            }
        }
        var e1=new Example();
        var e2=new Example();
    </script>
</body>
</html>

```

ex:2

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <script>
            class Example
            {
                constructor(id,name,sal)
                {
                    document.writeln("Employee Id :" + id + "<br>");
                    document.writeln("Employee Name :" + name + "<br>");
                    document.writeln("Employee Salary :" + sal + "<br>");
                }
            }
            var e1=new Example(101,'Alan',10000);
            var e2=new Example(102,'Jose',20000);
        </script>
    </body>
</html>

```

Abstraction

Hiding internal implementation and highlighting the set of services is called abstraction.
The best example of abstraction is GUI ATM machine where bank people will hide internal implementation and highlights the set of services like banking,withdrawl,mini stmt and etc.

Encapsulation

The process of encapsulating or grouping properties and it's associate functions is called encapsulation.

In encapsulation for every property we need to declare setter and getter methods.

```

<!DOCTYPE html>
<html>

```

```

<head>
    <title>MyPage!</title>
</head>
<body>
    <script>
        class Example
        {
            //setter method
            setId(id)
            {
                this.id=id;
            }
            setName(name)
            {
                this.name=name;
            }
            setSalary(sal)
            {
                this.sal=sal
            }

            //getter methods
            getId()
            {
                return this.id;
            }
            getName()
            {
                return this.name;
            }
            getSalary()
            {
                return this.sal;
            }
        }
        var e=new Example();
        e.setId(201);
        e.setName("Lara");
        e.setSalary(10000);
        document.writeln("Employee Id :" +e.getId()+"<br>");
        document.writeln("Employee Name :" +e.getName()+"<br>");
        document.writeln("Employee Salary :" +e.getSalary()+"<br>");
    </script>
</body>
</html>

```

ex:2

```

<!DOCTYPE html>
<html>
    <head>

```

```

<title>MyPage!</title>
</head>
<body>
<script>
    class Example
    {
        constructor(id,name,sal)
        {
            this.id=id;
            this.name=name;
            this.sal=sal;
        }
        //getter methods
        getId()
        {
            return this.id;
        }
        getName()
        {
            return this.name;
        }
        getSalary()
        {
            return this.sal;
        }
    }
    var e=new Example(301,"Nelson",20000);
    document.writeln("Employee Id :" +e.getId()+"<br>");
    document.writeln("Employee Name :" +e.getName()+"<br>");
    document.writeln("Employee Salary :" +e.getSalary()+"<br>");
</script>
</body>
</html>

```

Inheritance

Inheritance is a mechanism where one class will inherit the properties of another class.
Inheritance is a mechanism where one class will derive in the presence of another class.

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <script>
            class A
            {
                f1()
                {
                    document.writeln("A-F1 Function <br>");
                }
            }
        </script>
    </body>
</html>

```

```

        }
    class B extends A
    {
        f2()
        {
            document.writeln("B-F2 Function <br>");
        }
    }
    var a=new A();
    a.f1();
    var b=new B();
    b.f1();
    b.f2();

```

</script>

</body>

</html>

Polymorphism

Polymorphism has taken from Greek word.

Poly means many and morphism means forms.

The ability to represent in different forms is called polymorphism.

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
    </head>
    <body>
        <script>
            class A
            {
                display()
                {
                    document.writeln("A-class <br>");
                }
            }
            class B extends A
            {
                display()
                {
                    document.writeln("B-class <br>");
                }
            }
            var a=new A();
            a.display(); // A-class

            var b=new B();
            b.display(); // B-class

```

</script>

</body>

</html>

Bootstrap5

It is a most popular HTML, CSS and Javascript framework developed by twitter for developing web applications.

It is absolutely free to download and use.

It is developed by Mark Otto and Jacob Thornton at Twitter.

It is released as open source product on GITHUB in 2011.

Bootstrap contains HTML and CSS based designed templates for typography (fonts), forms, tables, list and etc.

It also contains javascript plugin's.

Bootstrap4 was released in the month of Augest 18, 2018.

Bootstrap5 which is extension of Bootstrap4 was release in May 5 , 2021.

Advantages of Bootstrap

It is easy to learn and easy to use.

It is used to develop responsive designs.

It saves lot of development time.

It is a consistent framework.

Good Documentation and Community support.

It supports greate Grid System.

It supports javascript plugins along with Jquery.

Boostrap color

By using utility classes change the color of a text.

```
<!DOCTYPE html>
<html>
<head>
    <title>mypage!</title>
    <!-- bootstrap cdn link -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
    <script
        src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>

</head>
<body>
    <h1 class="text-primary"> text primary </h1>
    <h1 class="text-secondary"> text secondary </h1>
    <h1 class="text-info"> text info </h1>
    <h1 class="text-warning"> text warning </h1>
    <h1 class="text-success"> text success </h1>
    <h1 class="text-danger"> text danger </h1>
    <h1 class="text-light"> text light </h1>
    <h1 class="text-dark"> text dark </h1>
</body>
</html>
```

```

ex:2
<!DOCTYPE html>
<html>
<head>
    <title>mypage!</title>
    <!-- bootstrap cdn link -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet">

    <script  src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"
></script>

    <style type="text/css">
        .text-brown
        {
            color:#B33771;
        }
    </style>

</head>
<body>
    <h1 class="text-primary"> text primary </h1>
    <h1 class="text-secondary"> text secondary </h1>
    <h1 class="text-info"> text info </h1>
    <h1 class="text-warning"> text warning </h1>
    <h1 class="text-success"> text success </h1>
    <h1 class="text-danger"> text danger </h1>
    <h1 class="text-light"> text light </h1>
    <h1 class="text-dark"> text dark </h1>
    <h1 class="text-brown"> text Brown </h1>
</body>
</html>

```

Bootstrap background

In bootstrap background color can be applied to any element/tag.

```

<!DOCTYPE html>
<html>
<head>
    <title>mypage!</title>
    <!-- bootstrap cdn link -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet">

```

```

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"
></script>
<style type="text/css">
    .bg-brown
    {
        background-color:#B33771;
    }
</style>
</head>
<body>
    <h1 class="bg-primary"> text primary </h1>
    <h1 class="bg-secondary"> text secondary </h1>
    <h1 class="bg-info"> text info </h1>
    <h1 class="bg-warning"> text warning </h1>
    <h1 class="bg-success"> text success </h1>
    <h1 class="bg-danger"> text danger </h1>
    <h1 class="bg-light"> text light </h1>
    <h1 class="bg-dark"> text dark </h1>
    <h1 class="bg-brown"> text Brown </h1>
</body>
</html>

```

Bootstrap border

Border utility is used to change quickly the border-style and border-radius of an element. It is mainly used for images, buttons and etc.

we can use border classes to an element to remove all borders or add some borders.

Add classes to an element to easily rounds its corners.

```

<!DOCTYPE html>
<html>
<head>
    <title>mypage!</title>
    <!-- bootstrap cdn link -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"
></script>
</head>
<body>
    <span class="border"> span1 tag</span>
    <span class="border-top"> span2 tag </span>
    <span class="border-start"> span3 tag </span>
    <span class="border-bottom"> span4 tag </span>
    <span class="border-end"> span5 tag </span>
</body>
</html>

```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>mypage!</title>
    <!-- bootstrap cdn link -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet">

    <script  src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"
></script>

</head>
<body>
    <span class="border border-primary"> span1 tag</span>
    <span class="border-top border-info"> span2 tag </span>
    <span class="border-start border-danger"> span3 tag </span>
    <span class="border-bottom border-success"> span4 tag </span>
    <span class="border-end border-dark"> span5 tag </span>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
<head>
    <title>mypage!</title>
    <!-- bootstrap cdn link -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet">

    <script  src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"
></script>

</head>
<body>
    
```

```









</body>
</html>

```

Bootstrap sizing

It is used to size the element by using sizing utility classes.
 we can easily make an element wide or a toll with our width and height utility.
 It suports for 25%,50%,75% and 100% bydefault.

```

<!DOCTYPE html>
<html>
<head>
  <title>MyPage!</title>

  <!-- add bootstrap plugins -->

  <meta charset="utf-8">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>

  <style>
    .w-60 {
      width:60%;
    }
  </style>

```

```

</head>
<body>
    <br>
    <div class="bg-primary w-25">Div1</div>
    <div class="bg-info w-50">Div2</div>
    <div class="bg-warning w-75">Div3</div>
    <div class="bg-success w-100">Div4</div>
    <div class="bg-danger w-60">Div5</div>
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>mypage!</title>
    <!-- bootstrap cdn link -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
    
    <br>
    
    <br>
    
    <br>
    
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>mypage!</title>
    <!-- bootstrap cdn link -->
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>

<style type="text/css">
    .w-60 {
        width:60%;
    }
    .h-60 {
        height:60%;
    }
</style>
</head>
<body>
    
    <br>
    
    <br>
    
    <br>
    
    <br>
    

</body>
</html>

```

Bootstrap containers

Containers are used to pad the content inside of them.

In bootstrap there are two containers classes available.

1. container-fluid
2. container

```

<!DOCTYPE html>
<html>
<head>
    <title>mypage!</title>
    <!-- bootstrap cdn link -->
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js">
</script>
</head>
<body>
    <div class="container-fluid bg-info">
        <h1>Heading Tag</h1>
        <p>Paragraph Tag</p>
    </div>
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<html>
<head>
    <title>mypage!</title>
    <!-- bootstrap cdn link -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js">
    </script>
</head>
<body>
    <div class="container bg-warning">
        <h1>Heading Tag</h1>
        <p>Paragraph Tag</p>
    </div>
</body>
</html>

```

Bootstrap5 Display

Bootstrap5 contains following list of properties.

ex: d-none
d-block
d-inline
d-inline-block

1) d-none

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">

```

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container">
        <h1 class="d-none"> This is Heading Tag </h1>
        <p> This is Paragraph Tag </p>
    </div>
</body>
</html>
```

2) d-block

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
    </head>
    <body>
        <div class="container">
            <span class="d-block bg-warning">This is span1</span>
            <span class="d-block bg-info">This is span2</span>
            <span class="d-block bg-primary">This is span3</span>
        </div>
    </body>
</html>
```

3) d-inline

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
    </head>
    <body>
        <div class="container">
            <div class="d-inline bg-primary">This is div1</div>
            <div class="d-inline bg-success">This is div2</div>
        </div>
    </body>
</html>
```

```

        <div class="d-inline bg-info">This is div3</div>
    </div>
</body>
</html>

4) d-inline-block
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
    </head>
    <body>
        <div class="container">
            <span class="bg-primary d-inline-block w-25">This is span1</span>
            <span class="bg-success d-inline-block w-25">This is span2</span>
            <span class="bg-info d-inline-block w-25">This is span3</span>
        </div>
    </body>
</html>

```

Bootstrap spacing

Spacing utility is used to assign the responsive friendly margin or padding values to an element or a subset of its sides with shorthand classes.

It includes individual properties , all properties ,vertical properties and horizontal properties.

bootstrap margin

m - for classes that set margin

Where sides is one of:

t - for classes that set margin-top or padding-top

b - for classes that set margin-bottom or padding-bottom

s - (start) for classes that set margin-left or padding-left in LTR, margin-right or padding-right in RTL

e - (end) for classes that set margin-right or padding-right in LTR, margin-left or padding-left in RTL

x - for classes that set both *-left and *-right

y - for classes that set both *-top and *-bottom

blank - for classes that set a margin or padding on all 4 sides of the element

Where size is one of:

0 - for classes that eliminate the margin or padding by setting it to 0

1 - (by default) for classes that set the margin or padding to \$spacer * .25

2 - (by default) for classes that set the margin or padding to \$spacer * .5

3 - (by default) for classes that set the margin or padding to \$spacer *

4 - (by default) for classes that set the margin or padding to \$spacer * 1.5

5 - (by default) for classes that set the margin or padding to \$spacer * 3
auto - for classes that set the margin to auto

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
    </head>
    <body>
        <div class="container">
            <h1 class="border border-primary m-1">This is heading 1</h1>
            <h1 class="border border-primary m-2">This is heading 2</h1>
            <h1 class="border border-primary m-3">This is heading 3</h1>
            <h1 class="border border-primary m-4">This is heading 4</h1>
            <h1 class="border border-primary m-5">This is heading 5</h1>
        </div>
    </body>
</html>
```

ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
    </head>
    <body>
        <div class="container-fluid">
            <h1 class="border border-primary mt-5">This is heading 1</h1>
            <h1 class="border border-primary mb-5">This is heading 2</h1>
            <h1 class="border border-primary ms-5">This is heading 3</h1>
            <h1 class="border border-primary me-5">This is heading 4</h1>
            <h1 class="border border-primary mx-5">This is heading 5</h1>
        </div>
    </body>
</html>
```

```

        <h1 class="border border-primary my-5">This is heading 6</h1>
    </div>
</body>
</html>

```

Bootstrap padding

p - for classes that set padding

Where sides is one of:

t - for classes that set margin-top or padding-top

b - for classes that set margin-bottom or padding-bottom

s - (start) for classes that set margin-left or padding-left in LTR, margin-right or padding-right in RTL

e - (end) for classes that set margin-right or padding-right in LTR, margin-left or padding-left in RTL

x - for classes that set both *-left and *-right

y - for classes that set both *-top and *-bottom

blank - for classes that set a margin or padding on all 4 sides of the element

Where size is one of:

0 - for classes that eliminate the margin or padding by setting it to 0

1 - (by default) for classes that set the margin or padding to \$spacer * .25

2 - (by default) for classes that set the margin or padding to \$spacer * .5

3 - (by default) for classes that set the margin or padding to \$spacer

4 - (by default) for classes that set the margin or padding to \$spacer * 1.5

5 - (by default) for classes that set the margin or padding to \$spacer * 3

auto - for classes that set the margin to auto

ex:1

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
    </head>
    <body>
        <div class="container-fluid">
            <h1 class="border border-primary p-1">This is heading 1</h1>
            <h1 class="border border-primary p-2">This is heading 2</h1>
            <h1 class="border border-primary p-3">This is heading 3</h1>
            <h1 class="border border-primary p-4">This is heading 4</h1>
            <h1 class="border border-primary p-5">This is heading 5</h1>
        </div>
    </body>
</html>

```

```

ex:2
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
    </head>
    <body>
        <div class="container-fluid">
            <h1 class="border border-primary pt-5">This is heading 1</h1>
            <h1 class="border border-primary pb-5">This is heading 2</h1>
            <h1 class="border border-primary ps-5">This is heading 3</h1>
            <h1 class="border border-primary pe-5">This is heading 4</h1>
            <h1 class="border border-primary px-5">This is heading 5</h1>
            <h1 class="border border-primary py-5">This is heading 6</h1>
        </div>
    </body>
</html>

```

Bootstrap 5 Grid System

Bootstrap's grid system is built with flexboxes.

To create a row in a grid system we need to use "row" class.

It allows to create 12 columns in a single row and It is not required that we must use all 12 columns.

The Grid system is responsive and the column will rearrange automatically dependent upon the screen size.

Grid class

According Bootstrap5 ,A grid system contains 6 classes.

col-	- extra small device
col-sm-	- small devices
col-md-	- medium devices
col-lg-	- large devices
col-xl-	- xlarge devices.
col-xxl-	- too xlarge device

ex:1

```

<!DOCTYPE html>
<html>
    <head>
        <title>IHUB TALENT</title>
        <!-- add bootstrap 5 plugins -->
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">

```

```

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container-fluid mt-5">
        <div class="row">
            <!-- 1 column occupies 12 column space -->
            <div class="col bg-primary">column1</div>
        </div>
    </div>
</body>
</html>

```

ex:2

```

<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>
    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container-fluid mt-5">
        <div class="row">
            <!-- 2 columns occupies 6 column space -->
            <div class="col bg-primary">column1</div>
            <div class="col bg-info">column2</div>
        </div>
    </div>
</body>
</html>

```

ex:3

```

<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>
    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

```

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
    <div class="container-fluid mt-5">
        <div class="row">
            <!-- 3 columns occupies 4 column space -->
            <div class="col bg-primary">column1</div>
            <div class="col bg-info">column2</div>
            <div class="col bg-warning">column3</div>
        </div>
    </div>
</body>
</html>
```

ex:4

```
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>
    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
    <div class="container-fluid mt-5">
        <div class="row">
            <!-- 4 columns occupies 3 column space -->
            <div class="col bg-primary">column1</div>
            <div class="col bg-info">column2</div>
            <div class="col bg-warning">column3</div>
            <div class="col bg-success">column4</div>
        </div>
    </div>
</body>
</html>
```

ex:5

```
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>
    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

</head>
<body>
    <div class="container-fluid mt-5">

        <div class="row">
            <!-- 5 columns occupies 3 column space -->
            <div class="col bg-primary">column1</div>

            <div class="col bg-info">column2</div>

            <div class="col bg-warning">column3</div>

            <div class="col bg-success">column4</div>

            <div class="col bg-danger">column5</div>
        </div>

    </div>
</body>
</html>

```

ex:6

```

<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

</head>
<body>
    <div class="container-fluid mt-5">

```

```

<div class="row">
    <div class="col-2 bg-primary">column1</div>
    <div class="col-4 bg-info">column2</div>
    <div class="col-6 bg-warning">column3</div>
</div>
</div>
</body>
</html>

```

ex:7

```

<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>
    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container-fluid mt-5">
        <div class="row">
            <div class="col-lg-4 bg-primary">column1</div>
            <div class="col-lg-4 bg-info">column2</div>
            <div class="col-lg-4 bg-warning">column3</div>
        </div>
        <br><br>
        <div class="row">
            <div class="col-lg-4 col-md-4 bg-primary">column1</div>
            <div class="col-lg-4 col-md-4 bg-info">column2</div>
            <div class="col-lg-4 col-md-4 bg-warning">column3</div>
        </div>
        <br><br>
        <div class="row">
            <div class="col-lg-4 col-md-4 col-sm-4 bg-primary">column1</div>

```

```

info">column2</div>
      <div class="col-lg-4 col-md-4 col-sm-4 bg-
warning">column3</div>
      </div>

      <br><br>
      <div class="row">
        <div class="col-lg-4 col-md-4 col-sm-4 col-12 bg-
primary">column1</div>
        <div class="col-lg-4 col-md-4 col-sm-4 col-12 bg-
info">column2</div>
        <div class="col-lg-4 col-md-4 col-sm-4 col-12 bg-
warning">column3</div>
      </div>

    </div>
</body>

</html>

```

Bootstrap images

3 classes to display the images.

- 1)rounded image (rounded)
- 2)circle image (rounded-circle)
- 3)thumbnail (img-thumbnail)

1)rounded image

```

<!DOCTYPE html>
<html>
<head>
  <title>IHUB TALENT</title>

  <!-- add bootstrap 5 plugins -->
  <meta charset="utf-8">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
  <div class="container-fluid mt-5">
    
  </div>
</body>
</html>

```

2)circle image

```
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

</head>
<body>
    <div class="container-fluid mt-5">
        
    </div>
</body>
</html>
```

3)thumbnail image

```
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

</head>
<body>
    <div class="container-fluid mt-5">
        
    </div>
</body>
</html>
```

Bootstrap float

Bootstrap float is used for alignment of an element.

ex:1

```
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>
    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container-fluid mt-5">
        
        
    </div>
</body>
</html>
```

ex:2

```
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>
    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container-fluid mt-5">
        
    </div>
</body>
</html>
```

Bootstrap Design

```
<!DOCTYPE html>
<html>
<head>
```

```

<title>IHUB TALENT</title>
<!-- add bootstrap 5 plugins -->
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
    <div class="container-fluid mt-5">
        <div class="row">
            <div class="col-lg-4 col-md-4 col-sm-6 col-12">
                
            </div>
            <div class="col-lg-4 col-md-4 col-sm-6 col-12">
                
            </div>
            <div class="col-lg-4 col-md-4 col-sm-6 col-12">
                
            </div>
        </div>
    </div>
</body>
</html>

```

Bootstrap tables

Tables are used to store the data in the form of rows and columns.

A "table" class is used to represent a table in bootstrap.

```

<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
    <div class="container mt-5">
        <table class="table">

```

```

<tr>
<th>ID</th>
<th>NAME</th>
<th>ADDRESS</th>
</tr>
<tr>
<td>101</td>
<td>Alan</td>
<td>USA</td>
</tr>
<tr>
<td>102</td>
<td>Jose</td>
<td>UAE</td>
</tr>
<tr>
<td>103</td>
<td>Nelson</td>
<td>UK</td>
</tr>
</table>
</div>

```

```

</body>
</html>

```

ex:

```

<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" >

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
<div class="container mt-5">

<table class="table table-bordered">
<tr>
<th>ID</th>
<th>NAME</th>
<th>ADDRESS</th>

```

```

        </tr>
        <tr>
            <td>101</td>
            <td>Alan</td>
            <td>USA</td>
        </tr>
        <tr>
            <td>102</td>
            <td>Jose</td>
            <td>UAE</td>
        </tr>
        <tr>
            <td>103</td>
            <td>Nelson</td>
            <td>UK</td>
        </tr>
    </table>

</div>

```

```

</body>
</html>

```

ex:

```

<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container mt-5">

        <table class="table table-borderless">
            <tr>
                <th>ID</th>
                <th>NAME</th>
                <th>ADDRESS</th>
            </tr>
            <tr>
                <td>101</td>
                <td>Alan</td>

```

```

        <td>USA</td>
    </tr>
    <tr>
        <td>102</td>
        <td>Jose</td>
        <td>UAE</td>
    </tr>
    <tr>
        <td>103</td>
        <td>Nelson</td>
        <td>UK</td>
    </tr>
</table>

</div>

</body>
</html>
```

ex:

```

<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" >
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container mt-5">
        <table class="table table-bordered table-striped">
            <tr>
                <th>ID</th>
                <th>NAME</th>
                <th>ADDRESS</th>
            </tr>
            <tr>
                <td>101</td>
                <td>Alan</td>
                <td>USA</td>
            </tr>
            <tr>
                <td>102</td>
```

```

        <td>Jose</td>
        <td>UAE</td>
    </tr>
    <tr>
        <td>103</td>
        <td>Nelson</td>
        <td>UK</td>
    </tr>
</table>

</div>

</body>
</html>
```

ex:

```

<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container mt-5">

        <table class="table table-bordered table-hover">
            <tr>
                <th>ID</th>
                <th>NAME</th>
                <th>ADDRESS</th>
            </tr>
            <tr>
                <td>101</td>
                <td>Alan</td>
                <td>USA</td>
            </tr>
            <tr>
                <td>102</td>
                <td>Jose</td>
                <td>UAE</td>
            </tr>
            <tr>
```

```

        <td>103</td>
        <td>Nelson</td>
        <td>UK</td>
    </tr>
</table>

</div>

</body>
</html>
```

Bootstrap list

It is used to represent list of items.

A list-group is applicable for unorder list.

A list-group-item is applicable for list of items.

```

<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container mt-5">
        Courses:
        <ul class="list-group">
            <li class="list-group-item">HTML</li>
            <li class="list-group-item">CSS</li>
            <li class="list-group-item">JavaScript</li>
            <li class="list-group-item">Bootstrap</li>
        </ul>
    </div>

```

```

</body>
</html>
```

ex:

```

<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" >

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container mt-5">
        Courses:
        <ul class="list-group">
            <li class="list-group-item active">HTML</li>
            <li class="list-group-item">CSS</li>
            <li class="list-group-item">JavaScript</li>
            <li class="list-group-item disabled">Bootstrap</li>
        </ul>
    </div>
</body>
</html>

```

ex:

```

<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" >

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container mt-5">
        Courses:
        <ul class="list-group list-group-flush">
            <li class="list-group-item ">HTML</li>
            <li class="list-group-item">CSS</li>
            <li class="list-group-item">JavaScript</li>
            <li class="list-group-item ">Bootstrap</li>
        </ul>
    </div>
</body>

```

```
</body>
</html>
```

Bootstrap Badges

Badges can be used as part of links or buttons to provide a counter.

ex:

```
<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" >
```

```
        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container mt-5">

        <span class="badge bg-primary">2</span> Notification
        <span class="badge bg-info">5</span> Email
        <span class="badge bg-warning">1K</span> Subscribes
        <span class="badge bg-success">1M</span> Likes

    </div>
```

```
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" >
```

```
        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
```

```

<div class="container mt-5">
    <span class="badge bg-primary">Advanced</span> Java
    <span class="badge bg-info">Basics</span> Oracle
    <span class="badge bg-warning">Intermediate</span> ReactJS
    <span class="badge bg-success">Optional</span> Spring Boot
</div>

</body>
</html>

```

Bootstrap alerts

A "alert" class is used to represent bootstrap alert.
We can use alert class along with contextual classes.

ex: alert-danger
 alert-info
 alert-success
 alert-warning
 and etc.

```

<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container mt-5">

        <div class="alert alert-danger">
            <strong>Sorry!</strong> Data is deleted
        </div>

        <div class="alert alert-success">
            <strong>Yahoo!</strong> Account is created
        </div>

        <div class="alert alert-warning">
            <strong>Listen!</strong> Don't learn fastly
        </div>
    </div>

```

```

<div class="alert alert-info">
    <strong>God!</strong> Very difficult
</div>

</div>

</body>
</html>

ex:
<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container mt-5">

        <div class="alert alert-danger collapse">
            <strong>Sorry!</strong> Data is deleted
        </div>

        <div class="alert alert-success">
            <strong>Yahoo!</strong> Account is created
        </div>

        <div class="alert alert-warning">
            <strong>Listen!</strong> Don't learn fastly
        </div>

        <div class="alert alert-info">
            <strong>God!</strong> Very difficult
        </div>

    </div>

</body>
</html>
```

```

ex:
<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container mt-5">

        <div class="alert alert-danger alert-dismissible">
            <strong>Sorry!</strong> Data is deleted
            <span class="btn-close" data-bs-dismiss="alert"></span>
        </div>

    </div>

</body>
</html>

```

Bootstrap buttons

The .btn class is intended to be used in conjunction with our button variants, or to serve as a basis for your own custom styles.

We need to use btn along with contextual classes

ex:

- btn-primary
- btn-info
- btn-danger
- and etc.

A .btn-outline-* ones to remove all background images and colors on any button.

```

<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

```

```

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" >

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
<div class="container mt-5">

    <button class="btn btn-primary"> button1 </button>
    <button class="btn btn-secondary"> button2 </button>
    <button class="btn btn-info"> button3 </button>
    <button class="btn btn-danger"> button4 </button>
    <button class="btn btn-warning"> button5 </button>
    <button class="btn btn-success"> button6 </button>

</div>

</body>
</html>

```

ex:

```

---
<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" >

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
<div class="container mt-5">

    <button class="btn btn-outline-primary"> button1 </button>
    <button class="btn btn-outline-secondary"> button2 </button>
    <button class="btn btn-outline-info"> button3 </button>
    <button class="btn btn-outline-danger"> button4 </button>
    <button class="btn btn-outline-warning"> button5 </button>
    <button class="btn btn-outline-success"> button6 </button>

</div>

</body>
</html>

```

```
</div>

</body>
</html>

ex:
<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container mt-5">

        <a href="" class="btn btn-outline-primary">clickME</a>

    </div>

</body>
</html>

ex:
<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
</head>
<body>
    <div class="container mt-5">
```

```
<button class="btn btn-link"> click </button>

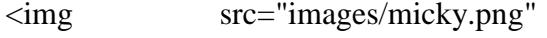
</div>

</body>
</html>
```

Bootstrap Design

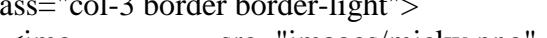
```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
    <style type="text/css">
      body
      {
        height: 100vh;
        display: flex;
        justify-content: center;
        align-items: center;
        background: linear-gradient(yellow,red);
      }
      .mystyle
      {
        box-shadow: 2px 2px 16px 6px #FFF;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <div class="mystyle">
        <div class="row ">
          <div class="col-3 border border-light">
```

```

class="mt-5 w-75 h-75 d-block m-auto"/>


```

```

class="mt-5 w-75 h-75 d-block m-auto"/>
</div>
<div class="col-3 border border-light">


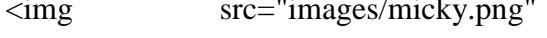
```

```

class="mt-5 w-75 h-75 d-block m-auto"/>
</div>
<div class="col-3 border border-light">


```

```

class="mt-5 w-75 h-75 d-block m-auto"/>
</div>
<div class="col-3 border border-light">


```

```

class="mt-5 w-75 h-75 d-block m-auto"/>
</div>
</div>

</div>

```

```

</body>
</html>

```

Bootstrap Glyphicons

Bootstrap includes 260 glyphicons from the Glyphicon Halflings set.

Glyphicons Halflings are normally not available for free, but their creator has made them available for Bootstrap free of cost.

We will see bootstrap glyphicons to display the icons in a web page.

```

<!DOCTYPE html>
<html>
  <head>
    <title>MyPage!</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
    <!-- Bootstrap CDN LINK for Icons -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css">

```

```
</head>
<body>

    <div class="container mt-5">
        <span class="bi bi-heart text-danger"></span>
    </div>
</body>
</html>
```

ex:2

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>

        <meta charset="utf-8">

        <meta name="viewport" content="width=device-width, initial-scale=1">

        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">

        <script
            src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>

        <!-- Bootstrap CDN LINK for Icons -->
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css">

    </head>
    <body>

        <div class="container mt-5">
            <button class="bi bi-search btn btn-primary" type="button">Search</button>
        </div>
    </body>
</html>
```

ex:3

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>

        <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>

<!-- Bootstrap CDN LINK for Icons -->
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css">

</head>
<body>

<div class="container mt-5">

    <span class="bi bi-facebook text-primary"> </span>
    <span class="bi bi-twitter"> </span>
    <span class="bi bi-instagram"> </span>
    <span class="bi bi-youtube"> </span>
    <span class="bi bi-whatsapp text-success"> </span>

</div>
</body>
</html>

```

Bootstrap Cards

A card in Bootstrap 5 is a bordered box with some padding around its content. It includes options for headers, footers, content, colors, etc.

A basic card is created with the .card class, and content inside the card has a .card-body class: The .card-header class adds a heading to the card and the .card-footer class adds a footer to the card.

Use .card-title to add card titles to any heading element.

The .card-text class is used to remove bottom margins for a <p> element if it is the last child (or the only one) inside .card-body.

Add .card-img-top to an to place the image at the top inside the card.

Note that we have added the image outside of the .card-body to span the entire width:

```

<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

```

```

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

<!-- Bootstrap Font Icon CSS -->
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css">

</head>
<body>
    <div class="container mt-5">

        <div class="card">
            <div class="card-header">
                Heading
            </div>
            <div class="card-body">
                Body
            </div>
            <div class="card-footer">
                Footing
            </div>
        </div>

    </div>
</body>
</html>

```

ex:2

```

-----
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

    <!-- Bootstrap Font Icon CSS -->

```

```

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.5.0/font/bootstrap-icons.css">

</head>
<body>
    <div class="container mt-5">

        <div class="card">
            <div class="card-header">
                Heading
            </div>
            <div class="card-body">
                <div class="card-title">Bootstrap</div>
                <div class="card-text">Bootstrap is a css
framework</div>
            </div>
            <div class="card-footer">
                Footing
            </div>
        </div>

    </div>
</body>
</html>

```

ex:3

```

<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet">
        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" ></script>
        <!-- Bootstrap CDN LINK for Icons -->
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.5.0/font/bootstrap-icons.css">
    </head>
    <body>
        <div class="container mt-5">

```

```
<div class="row">

    <div class="col-4">
        <div class="card">
            
            <div class="card-body">
                <h2 class="card-title" style="text-align: center;">
                    Thumbnail-1</h2>
                <p class="card-text" style="text-align: center;">
                    Illustration</p>
                </div>
            </div>
        </div>
    </div>

    <div class="col-4">
        <div class="card">
            
            <div class="card-body">
                <h2 class="card-title" style="text-align: center;">
                    Thumbnail-2</h2>
                <p class="card-text" style="text-align: center;">
                    Illustration</p>
                </div>
            </div>
        </div>
    </div>

    <div class="col-4">
        <div class="card">
            
            <div class="card-body">
                <h2 class="card-title" style="text-align: center;">
                    Thumbnail-3</h2>
                <p class="card-text" style="text-align: center;">
                    Illustration</p>
                </div>
            </div>
        </div>
    </div>

</div>
</body>
</html>
```

Bootstrap Collapse Plugin

Collapsibles are useful when you want to hide and show large amount of content:

A `data-bs-toggle="collapse"` attribute is used to show and hide the content.

A `data-bs-target` attribute is used to connect a button with div tag.

A "collapse" class inside `<div>` is used to hide/collapse the content for first time.

ex:1

```
---  
<!DOCTYPE html>  
<html>  
<head>  
    <title>IHUB TALENT</title>  
  
    <!-- add bootstrap 5 plugins -->  
    <meta charset="utf-8">  
  
    <meta name="viewport" content="width=device-width, initial-scale=1">  
  
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >  
  
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>  
  
    <!-- Bootstrap Font Icon CSS -->  
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css">  
  
</head>  
<body>  
    <div class="container mt-5">  
        <button class="btn btn-primary" data-bs-toggle="collapse" data-bs-target="#myId"> Toggle </button>  
        <h1 id="myId" class="collapse">This is heading tag</h1>  
    </div>  
</body>  
</html>
```

ex:2

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>IHUB TALENT</title>
```

```

<!-- add bootstrap 5 plugins -->
<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

<!-- Bootstrap Font Icon CSS -->
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css">

</head>
<body>
    <div class="container mt-5">
        <a href="#myId" data-bs-toggle="collapse"> Toggle </a>
        <h1 id="myId" class="collapse">This is heading tag</h1>
    </div>
</body>
</html>

```

Bootstrap collapse plugin with cards

```

<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

    <!-- Bootstrap Font Icon CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css">

```

```

</head>
<body>
    <div class="container mt-5">
        <div class="card">
            <div class="card-header">
                <button class="btn btn-outline-primary"
                    data-bs-toggle="collapse"
                    data-bs-target="#myId"> Toggle
            </button>
            </div>
            <div class="card-body collapse" id="myId">
                <div class="row">
                    <div class="col-4">
                        
                    </div>
                    <div class="col-4">
                        
                    </div>
                    <div class="col-4">
                        
                    </div>
                </div>
            </div>
        </div>
    </div>
</body>
</html>

```

Bootstrap responsive navbar

Navbars require a wrapping .navbar with .navbar-expand{-sm|-md|-lg|-xl|-xxl} for responsive collapsing and color scheme classes.

Navbars and their contents are fluid by default.

Change the container to limit their horizontal width in different ways.

Use our spacing and flex utility classes for controlling spacing and alignment within navbars.

Navbars are responsive by default, but you can easily modify them to change that.

Responsive behavior depends on our Collapse JavaScript plugin.

.navbar-brand for your company, product, or project name.

.navbar-nav for a full-height and lightweight navigation (including support for dropdowns).

.navbar-toggler for use with our collapse plugin and other navigation toggling behaviors.

Flex and spacing utilities for any form controls and actions.

.navbar-text for adding vertically centered strings of text.

.collapse.navbar-collapse for grouping and hiding navbar contents by a parent breakpoint.

Add an optional .navbar-scroll to set a max-height and scroll expanded navbar content.

```
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

    <!-- Bootstrap Font Icon CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css">

</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">
            <a class="navbar-brand" href="#">IHUBTALENT</a>

            <button class="navbar-toggler" data-bs-toggle="collapse" data-bs-target="#navbarId">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarId">
                <ul class="navbar-nav">
                    <li class="nav-item">
                        <a class="nav-link active" href="#">Home</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" href="#">About</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" href="#">Service</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" href="#">Portfolio</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" href="#">Contact</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>

```

```

        </li>
    </ul>
    </div>
</div>
</nav>
</body>
</html>

ex:2
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

    <!-- Bootstrap Font Icon CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css">

</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container-fluid">
            <a class="navbar-brand" href="#">IHUBTALENT</a>

            <button class="navbar-toggler" data-bs-toggle="collapse" data-bs-target="#navbarId">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarId">
                <ul class="navbar-nav ">
                    <li class="nav-item">
                        <a class="nav-link active" href="#">Home</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" href="#">About</a>
                    </li>
                    <li class="nav-item">

```

```

        <a class="nav-link active" href="#">Service</a>
    </li>
    <li class="nav-item">
        <a class="nav-link active" href="#">Portfolio</a>
    </li>
    <li class="nav-item">
        <a class="nav-link active" href="#">Contact</a>
    </li>
</ul>
</div>
</div>
</nav>
</body>
</html>

```

ex:3

```

<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

    <!-- Bootstrap Font Icon CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css" >

</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
        <div class="container-fluid">
            <a class="navbar-brand" href="#">IHUBTALENT</a>

            <button class="navbar-toggler"
                    data-bs-toggle="collapse"
                    data-bs-target="#navbarId">

                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarId">
                <ul class="navbar-nav " >

```

Bootstrap Scrollspy

The scrollspy plugin automatically highlights the navigation links based on the scroll position to indicate where the user is currently on the page.

```
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>
    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js"></script>

<!-- Bootstrap Font Icon CSS -->
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css">

<style>
    .section1
    {
        width: 100%;
        height: 560px;
        background-color: #C4E538;
    }
    .section2
    {
        width: 100%;
        height: 560px;
        background-color: #ED4C67;
    }
    .section3
    {
        width: 100%;
        height: 560px;
        background-color: #9980FA;
    }
    .section4
    {
        width: 100%;
        height: 560px;
        background-color: #12CBC4;
    }
    .section5
    {
        width: 100%;
        height: 560px;
        background-color: #833471;
    }
</style>

</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
        <div class="container-fluid">
```

```

<a class="navbar-brand" href="#">IHUBTALENT</a>

<button class="navbar-toggler"
       data-bs-toggle="collapse"
       data-bs-target="#navbarId">

    <span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarId">
    <ul class="navbar-nav ">
        <li class="nav-item">
            <a class="nav-link active" href="#homeId">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link active" href="#aboutId">About</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="#serviceId">Service</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="#portfolioId">Portfolio</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="#contactId">Contact</a>
        </li>
    </ul>
    </div>
</div>
</nav>

<section class="section1" id="homeId">
    <h1 class="pt-5 text-center">
        Home Page
    </h1>
</section>
<section class="section2" id="aboutId">
    <h1 class="pt-5 text-center">
        About Page
    </h1>
</section>
<section class="section3" id="serviceId">
    <h1 class="pt-5 text-center">
        Service Page
    </h1>
</section>
<section class="section4" id="portfolioId">
    <h1 class="pt-5 text-center">

```

```

        Portfolio Page
    </h1>
</section>
<section class="section5" id="contactId">
    <h1 class="pt-5 text-center">
        Contact Page
    </h1>
</section>

</body>
</html>

```

Bootstrap5 forms

Bootstrap 5 supports two types of form layouts.

- 1) Stacked Forms
- 2) Inline forms

1) Stacked Forms

All textual `<input>` and `<textarea>` elements with class `.form-control` get proper form styling. we add a `.form-label` class to each label element to ensure correct padding. Checkboxes have different markup. They are wrapped around a container element with `.form-check`, and labels have a class of `.form-check-label`, while checkboxes and radio buttons use `.form-check-input`.

```

<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

    <!-- Bootstrap Font Icon CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css">
</head>
<body>
    <div class="container mt-5">

        <!-- stack form -->
        <form action="">

            <label class="my-3">UserName:</label>

```

```

        <input type="text" name="t1" class="form-control"
placeholder="username" autocomplete="off" />

        <label class="my-3">Password:</label>
        <input type="password" name="t1" class="form-control"
placeholder="password" />

        <input type="submit" value="submit" class="btn btn-primary
my-5 w-100"/>

    </form>

```

```

</div>
</body>
</html>
```

ex:2

```

<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

    <!-- Bootstrap Font Icon CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css" >
</head>
<body>

    <div class="container mt-5">

        <div class="w-50">
        <!-- stack form -->
        <form action="">

            <label class="my-3">UserName:</label>
            <input type="text" name="t1" class="form-control"
placeholder="username" autocomplete="off" />
```

```
<label class="my-3">Password:</label>
<input type="password" name="t1" class="form-control"
placeholder="password" />

<input type="submit" value="submit" class="btn btn-primary
my-5 w-100"/>

</form>
</div>
```

```
</div>
</body>
</html>
```

ex:3

```
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

    <!-- Bootstrap Font Icon CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css" >
</head>
<body>
    <div class="container mt-5">

        <form action="">
            <label>Address:</label>
            <textarea name="t1" rows="5" cols="10" class="form-control"></textarea>
        </form>

    </div>
</body>
</html>
```

```

ex:4
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

    <!-- Bootstrap Font Icon CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css">
</head>
<body>
    <div class="container mt-5">

        <form action="">

            <div class="form-check">
                <input type="radio" name="t1" value="male" class="form-check-input"/>
                <label class="form-check-label">MALE</label>
            </div>

            <div class="form-check">
                <input type="radio" name="t1" value="female" class="form-check-input"/>
                <label class="form-check-label">FEMALE</label>
            </div>

        </form>

    </div>
</body>
</html>

```

```

ex:5
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->

```

```

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

<!-- Bootstrap Font Icon CSS -->
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css">
</head>
<body>
    <div class="container mt-5">

        <form action="">

            <div class="form-check">
                <input type="checkbox" name="t1" value="married" class="form-check-input"/>
                <label class="form-check-label">MARRIED</label>
            </div>

            <div class="form-check">
                <input type="checkbox" name="t1" value="single" class="form-check-input"/>
                <label class="form-check-label">SINGLE</label>
            </div>

        </form>

    </div>
</body>
</html>

ex:6
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

```

```

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

<!-- Bootstrap Font Icon CSS -->
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css">
</head>
<body>
    <div class="container mt-5">

        <form action="">

            <label>Select Country:</label>
            <select name="t1" class="form-select">
                <option value="">none</option>
                <option value="india">India</option>
                <option value="america">America</option>
                <option value="australia">Australia</option>
                <option value="canada">Canada</option>
            </select>

        </form>

    </div>
</body>
</html>

```

ex:7

```

<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

    <!-- Bootstrap Font Icon CSS -->

```

```

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.5.0/font/bootstrap-icons.css">
</head>
<body>
    <div class="container mt-5">

        <form action="">
            <label>Performance Rating :</label>
            <input type="range" name="t1" class="form-range"/>
        </form>

    </div>
</body>
</html>

ex:8
<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/js/bootstrap.bundle.min.js" ></script>

    <!-- Bootstrap Font Icon CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.5.0/font/bootstrap-icons.css">
</head>
<body>
    <div class="container mt-5">

        <form action="">
            <div class="input-group">
                <label class="input-group-text">www.</label>
                <input type="text" name="t1"/>
                <label class="input-group-text">.com</label>
            </div>
        </form>
    </div>
</body>
</html>

```

```

        </div>

    </form>

</div>
</body>
</html>
```

2. Inline Forms

If you want your form elements to appear side by side, use .row and .col.

```

<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

    <!-- Bootstrap Font Icon CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css" >
</head>
<body>
    <div class="container mt-5">

        <form action="">

            <div class="row">

                <div class="col-2">
                    <label>UserName:</label>
                </div>
                <div class="col-3">
                    <input type="text" name="t1"/>
                </div>
                <div class="col-2">
                    <label>Password:</label>
                </div>
                <div class="col-3">
                    <input type="password" name="t1"/>
                </div>
            </div>
        </form>
    </div>
</body>
```

```

        </div>
    <div class="col-2">
        <input type="submit" value="submit"/>
    </div>
</div>

</form>

</div>
</body>
</html>

```

Bootstrap Carousel

A .carousel class creates a carousel

A .slide class adds a CSS transition and animation effect when sliding from one item to the next. Remove this class if you do not want this effect.

A .carousel-inner class adds slides to the carousel.

A .carousel-item class specifies the content of each slide.

Add elements inside <div class="carousel-caption"> within each <div class="carousel-item"> to create a caption for each slide.

A .carousel-indicators class adds indicators for the carousel.

These are the little dots at the bottom of each slide (which indicates how many slides there are in the carousel, and which slide the user are currently viewing).

A .carousel-control-prev class adds a left (previous) button to the carousel, which allows the user to go back between the slides.

A .carousel-control-next class adds a right (next) button to the carousel, which allows the user to go forward between the slides.

A .carousel-control-prev-icon class used together with .carousel-control-prev to create a "previous" button.

A .carousel-control-next-icon class used together with .carousel-control-next to create a "next" button.

```

<!DOCTYPE html>
<html>
<head>
    <title>IHUB TALENT</title>

    <!-- add bootstrap 5 plugins -->
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>

    <!-- Bootstrap Font Icon CSS -->

```

```

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.5.0/font/bootstrap-icons.css">
</head>
<body>
    <div class="container mt-5">

        <div id="carouselId" class="carousel slide" data-bs-ride="carousel">
            <div class="carousel-indicators">
                <button type="button" data-bs-target="#carouselId" data-bs-slide-to="0" class="active"></button>
                <button type="button" data-bs-target="#carouselId" data-bs-slide-to="1"></button>
                <button type="button" data-bs-target="#carouselId" data-bs-slide-to="2"></button>
            </div>
            <div class="carousel-inner">
                <div class="carousel-item active">
                    
                    <div class="carousel-caption">
                        <h1>Washington DC</h1>
                        <p>Our Team Is Our Strength</p>
                    </div>
                </div>
                <div class="carousel-item">
                    
                    <div class="carousel-caption">
                        <h1>Los Angeles</h1>
                        <p>Our Work Is Our Identity</p>
                    </div>
                </div>
                <div class="carousel-item">

```

```
class="w-100 mx-auto d-block"/>


<div class="carousel-caption">
  <h1>New Jersey</h1>
  <p>Our Service are Best Practice</p>
</div>

</div>

</div>

<button class="carousel-control-prev" type="button" data-bs-target="#carouselId" data-bs-slide="prev">
  <span class="carousel-control-prev-icon"></span>
  <span class="visually-hidden">Previous</span>
</button>

<button class="carousel-control-next" type="button" data-bs-target="#carouselId" data-bs-slide="next">
  <span class="carousel-control-next-icon"></span>
  <span class="visually-hidden">Next</span>
</button>

</div>

</div>
</body>
</html>
```

React/ReactJS

It is a declarative, efficient and flexible javascript frontend library used to develop frontend applications and user interfaces i.e UI.

It is an open source, component based javascript frontend library responsible only for view layer of the application.

It is developed by Jordon Walke who was a software engineer at Facebook.

It is initially introduced by Facebook and later it is used in their own products like Instagram and facebook.

It was released to the public in the month of May, 2013.

The official website of reactjs is <https://react.dev>.

The latest version of reactjs is v18.2.2.

The main objective to introduced reactjs is to develop reusable components.

A component is a building block of react application.

Advantages of ReactJS

1. It is easy to learn and easy to use.
2. It is used to develop reusable components.
3. It supports one way data binding.
4. It supports virtual DOM.
5. Good documentation and Community support.
6. It supported by all major browsers.

Q) What is the difference between Angular and ReactJS ?

Angular

It is a product of Google.

It was released in Oct,2015.

It is a javascript framework which is used to develop web and mobile applications.

It supports traditional DOM.

It supports two way data binding.

It is used to develop rich featured Application.

TypeScript language is used.

Jasmine and Karma is used as a testing framework.

The default port number is 4200.

Angular used by Google, McDonalds, Nike and etc.

ReactJS

It is a product of Facebook.

It was released in May,2013.

It is a javascript open source library responsible only for view layer of the application.

It supports virtual DOM.

It supports one way data binding.

It is used to develop Single Page Applications (SPA)

JSX language is used.

Jest and Enzyme is used as a testing framework.

The default port number is 3000.

React used by Facebook , Instagram, youtube airbnb ,skype and etc.

Pre-Requisition to Learn ReactJS

- 1) Good Knowledge on HTML,CSS and JavaScript.
- 2) Idea on JSX
- 3) Basic of ES6.
- 4) npm commands

How ReactJS works?

ReactJS internally uses virtual DOM.

Virtual DOM is also a Tree Node Structure.

Virtual DOM will find a effective way to make the changes in traditional/real DOM. Hence react applications will execute fastly.

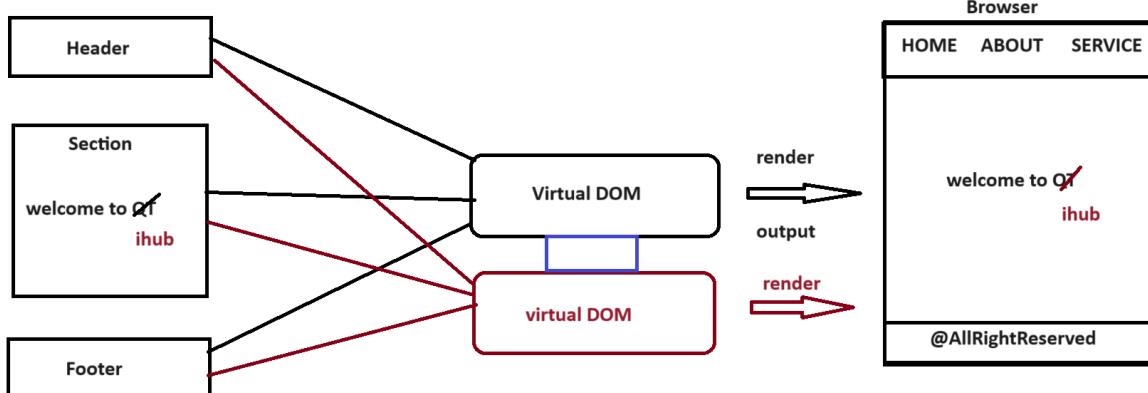


Diagram: react1.1

JSX

JSX stands for JavaScript XML.

JSX allows us to write HTML code in Javascript.

JSX element contains tags, attributes and childrens.

JSX is a not a neccessity to create react application instead we can use Babel.

JSX makes our program simpler and elegant.

JSX ultimately transpile to pure javascript which is understand by a browser window.

JSX elements

JSX elements allows us to write HTML code without using createElement() method or appendChild() method.

ex:1 JSX

```

<h1> Heading Tag </h1>
Babel
  React.createElement('h1',null,'Heading Tag');
  Here
  'h1' is a tag name
  'null' is a optional attribute
  'Heading Tag' is a children
  
```

ex:2 JSX

```

<div> <h1>Heading Tag</h1> </div>
Babel
  React.createElement('div',null,
    React.createElement('h1',null,'Heading Tag'));
  
```

ex:

JSX

```

<h1 id="myId"> Heading Tag </h1>
Babel
  React.createElement('h1',{id:'myId'},'Heading Tag');
  
```

ex:

JSX

```

<h1 className="myClass"> Heading Tag </h1>
Babel
  React.createElement('h1',{class:'myClass'},'Heading Tag');
  
```

ex: JSX

```

<h1 id="myId" className="myClass"> Heading Tag </h1>
  
```

Babel

```
React.createElement('h1',{id:'myId',class:'myClass'},'Heading Tag');
```

JSX Expression

JSX expression is used to represent expression in curly brace i.e. {}.

JSX expression can be variables, constants and any valid javascript expressions.

ex:1 var a=10;
 <h1>{a}</h1>

ex:2 <h1>{10 + 20 }</h1>

ex:3 <h1>{Math.random()}</h1>

npm

npm stands for Node Package Manager.

It is a integrated tool for nodejs.

npm is used to install node dependencies/packages/libraries.

We can install node dependencies as follow.

ex: npm install -g dependency_name/library/package_name

All the dependencies will be installed in "node_modules" folder.

Steps to work with npm

step1: Download and Install nodejs software.

ex: <https://nodejs.org/en>

step2: Copy nodejs directory.

ex: C:\Program Files\nodejs

step3: Paste nodejs directory in environmental variables.

ex: right click to my pc --> properties --> advanced system settings
 ---> environmental variables

 user variables --> click to new button -->

 variable name : path

 variable value :

 C:\Program Files\nodejs;C:\Users\Dell\AppData\Roaming\npm;

step4: Check the environmental setup done perfectly or not.

ex: cmd> node -v
 cmd> npm -v

step5: Install npm by using below command.

ex: cmd> npm install -g npm
 or
 cmd> npm install -g create-react-app

Steps to develop First React application

step1: Make sure Nodejs setup is done perfectly.

step2: Download and Install VSC (Visual Code Editor) editor.

ex: <https://code.visualstudio.com/>

step3: Create a "ReactProjects" folder inside "E" drive.

step4: Open the command prompt from "Reactprojects" folder.

step5: Open VSC editor from "Reactprojects" folder.

ex: Reactprojects> code .

step6: Create a react application/project i.e myapp1.
ex: Reactprojects>npx create-react-app myapp1

step7: Switch to the react project.
ex: Reactprojects> cd myapp1

step8: Run the react project by using below command.
ex: Reactprojects/myapp1> npm start

step9: Test the application by using below request url.
ex: http://localhost:3000

*Note: React application runs in a light weight development server with default 3000 port no.

Q) How to create react project /application ?

npx create-react-app myapp1

Q) How to move to the project in react?

cd myapp1

Q) How to run react application/ project?

npm start

React Project Structure and work flow

```
myapp1
|
|---node_modules
|
|---public
    |
    |---favicon.ico
    |---index.html
    |---manifest.json
|
|----src
    |
    |---index.js
    |---index.css
    |
    |---App.js
    |---App.css
    |
    |---App.test.js
|
|----package.json
|----README.md
```

A "node_modules" contains all dependencies and libraries installed.

A "favicon.ico" is a favrouite icon of a react application.

A "index.html" is a main tempate of react application.

A "manifest.json" file contains metadata which is used when we install application on client mobile or computer.

A "index.js" file is a entry point.

A "index.css" file is related to index.js and it is global.
A "App.js" is a parent component.
A "App.css" file is related to App.js and it is global.
A "App.test.js" file is related to unit testing.
A "package.json" file contains dependencies along with versions.

Note: index.html - main template
index.js - entry point
App.js - parent component
package.json - dependencies with versions

Work flow

code load to render to output
App.js -----> index.js -----> index.html -----> Browser

Steps to develop second application in react

- step1: create a react application i.e myapp2.
ex: ReactProjects> npx create-react-app myapp2
step2: Open the VSC editor.
ex: ReactProjects> code .
step3: Switch to myapp2 project.
ex: ReactProjects> cd myapp2
step4: Run the react application.
ex: ReactProjects/myapp2> npm start
step5: Test the application by using below request url.
ex: http://localhost:3000
step6: Write below code in App.js file.

Approach1

App.js
function App()
{
 return(
 <h1>I Love ReactJS </h1>
)
}
export default App;

Approach2

App.js
var App=function(){

 return(
 <h1>I Love ReactJS Programming</h1>
)
}
export default App;

Approach3

App.js

```
var App=()=>{  
  
    return(  
        <h1>I Love ReactJS Programming and Development</h1>  
    )  
}  
  
export default App;
```

*React is mainly used to develop reusable components.

App.js

```
function App() {  
    return (  
        <h1>  
            React Example for Reusability  
        </h1>  
    )  
}  
  
export default App
```

index.js

```
import React from 'react';  
import ReactDOM from 'react-dom/client';  
import './index.css';  
import App from './App';  
import reportWebVitals from './reportWebVitals';  
  
const root = ReactDOM.createRoot(document.getElementById('root'));  
root.render(  
    <React.StrictMode>  
        <App />  
        <App />  
        <App />  
    </React.StrictMode>  
);  
  
// If you want to start measuring performance in your app, pass a function  
// to log results (for example: reportWebVitals(console.log))  
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals  
reportWebVitals();
```

React Fragment

Fragment is used to group of list of childrens without adding extra nodes of the DOM.
In general, We can return only one element at a time but we can't return more then one element directly.

To return more then one element we need to use React Fragment.

syntax <React.Fragment>

-

-

```
</React.Fragment>
or
<>
-
-
</>
```

Examples

App.js

```
function App
{
    return (
        //return react element
        return <h1>IHUB Talent</h1>
        <h2>React Tutorial For Freshers</h2>
    );
}
//export React component
export default App
```

o/p: Filed to compile

To overcome above problem we can use `<div>` tag and inside that `<div>` tag we can declare any child tags.

ex:

```
App.js
function App
{
    return (
        //return react element
        return
            <div>
                <h1>IHUB Talent</h1>
                <h2>React Tutorial For Freshers</h2>
            </div>
    );
}
//export React component
export default App
```

Note: In above program "`<div>`" tag is a unused tag.

To remove unused/unnecessary tags we can use React Fragment.

approach1

App.js

```
import React from "react";
function App()
```

```

{
  return (
    <React.Fragment>
      <h1>IHUB React Tutorial</h1>
      <h1>React Classes for Freshers</h1>
    </React.Fragment>
  );
}

export default App;

```

approach2

App.js

```

import React from "react";
import { Fragment } from 'react';
function App()
{
  return (
    <Fragment>
      <h1>IHUB React Tutorial</h1>
      <h1>React Classes for Freshers</h1>
    </Fragment>
  );
}

export default App;

```

approach3

App.js

```

import React from "react";

function App()
{
  return (
    <>
      <h1>IHUB React Tutorial</h1>
      <h1>React Classes for Freshers</h1>
    </>
  );
}

export default App;

```

React Components

A component is a building block of react application.

Components allows us to split our UI into independent reusable pieces.

ex: <Header> , <Footer>, <Section>, <Table>, <Form> and etc.

React components are like javascript functions because they accept arbitrary inputs like props and return react element describing what should appear on the screen.

React component name always starts with uppercase letter.

There are two ways to declare react components.

- 1) Function Component / Functional Component
- 2) Class Component

1) Function Component

Function component is a javascript function which takes props as arguments along with inputs.

Function component is also known as stateless component because it does not hold state.

syntax:1

```
function App()
{
    return
    (
        <h1> Named Function </h1>
    )
}
export default App;
```

syntax:2

```
var App=function()
{
    return
    (
        <h1> Anonymous Function </h1>
    )
}
export default App;
```

syntax:3

```
var App=()=>
{
    return
    (
        <h1> Arrow Function </h1>
    )
}
export default App;
```

Project structure

```
myapp3
|
|--node_modules
|
|---public
    |
    |---manifest.json
    |---index.html
    |---favicon.ico
```

```
|  
|-----src  
|       |  
|       |---index.js  
|       |  
|       |---App.js
```

```
|-----package.json  
|-----README.md
```

step1: create a react application i.e myapp3.

ex: Reactprojects> npx create-react-app myapp3

step2: Open VSC editor from Reactprojects folder.

ex: Reactprojects> code .

step3: Jump/Switch to myapp3 project.

ex: Reactprojects> cd myapp3

step4: Run the react application.

ex: Reactprojects/myapp3> npm start

step5: Test the react application by using below request url.

ex: http://localhost:3000

step6: Declare below code inside App.js file.

```
App.js  
var App=()=>{  
  return (  
    <h1>Arrow Function component</h1>  
  )  
}  
export default App
```

Function component with props

In order to use props in a component We need to perform following changes in react "myapp3" project.

index.js

```
import React from 'react';  
import ReactDOM from 'react-dom/client';  
import App from './App';  
import reportWebVitals from './reportWebVitals';  
  
const root = ReactDOM.createRoot(document.getElementById('root'));  
root.render(  
  <React.StrictMode>  
    <App name="Alan" rollno="101"/>  
  </React.StrictMode>  
);  
  
// If you want to start measuring performance in your app, pass a function  
// to log results (for example: reportWebVitals(console.log))  
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
```

```
reportWebVitals();
```

App.js

```
var App=(props)=>{
  return (
    <>
    <h1>Name : {props.name}</h1>
    <h1>RollNo : {props.rollno}</h1>
    </>
  )
}
export default App
```

2) Class component

A class Component requires to extends from React Component.

The class must implements a render() method function which returns A react Element to be render.This is Similar to return value of a functional component.

In a class based component props are accessible via this.props.

The class component is also known as a stateful component because they can hold or manage local state.

Project structure

```
myapp4
|
|---node_modules
|
|----public
|      |
|      |---manifest.json
|      |---index.html
|      |---favicon.ico
|
|----src
|      |
|      |---index.js
|      |
|      |---App.js
|
|----package.json
|----README.md
```

step1: create a react application i.e myapp4.

ex: Reactprojects> npx create-react-app myapp4

step2: Open VSC editor from Reactprojects folder.

ex: Reactprojects> code .

step3: Jump/Switch to myapp4 project.

ex: Reactprojects> cd myapp4

step4: Run the react application.

ex: Reactprojects/myapp4> npm start

step5: Test the react application by using below request url.

ex: http://localhost:3000

step6: Declare below code inside App.js file.

ex:

App.js

```
import {Component} from 'react';
class App extends Component
{
    render()
    {
        return(
            <h1>Class Component</h1>
        )
    }
}
export default App
```

Class component with props

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App name="Jose" rollno="501"/>
  </React.StrictMode>
);
// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

App.js

```
import {Component} from 'react';
class App extends Component
{
    render()
    {
        return(
            <>
                <h1>Name :{this.props.name}</h1>
                <h1>RollNo :{this.props.rollno}</h1>
            </>
        )
    }
}
export default App
```

Composing Components in React

A component can refer to other components in their output is called composing component.

Let us use some component abstraction for any level of details.

Project structure

```
myapp4
|
|----node_modules
|
|----public
|    |
|    |---index.html (main template)
|    |---favicon.ico (favicon)
|    |---manifest.json (metadata)
|
|----src
|    |
|    |---index.js (entry point)
|    |
|    |
|    |---App.js (parent component)
|    |
|    |---Student.js (custom component)
|
|----package.json
|----README.md
```

step1: Create a React Application.

ex: ReactProjects>npx create-react-app myapp4

step2: Start Visual Studio Code (VSC) Editor.

ex: ReactProjects> code .

step3: Delete all the files from "src" folder.

step4: Create "index.js" file inside "src" folder.

index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";

const root=ReactDOM.createRoot(document.getElementById('root'));
```

```
root.render(
  <React.StrictMode>
    <App/>
  </React.StrictMode>
)
```

step5: Create App.js file inside "src" folder.

App.js

```
import Student from './Student';
```

```
function App()
{
  return (
    <Student/>
  )
}
export default App;
```

step6: Create Student.js file inside "src" folder.

Student.js

```
function Student()
{
  return (
    <h1>Student Component</h1>
  )
}
```

export default Student;

step7: Move to myapp4.

ex: ReactProjects> cd myapp4

step8: Run the react application.

ex: ReactProjects/myapp4> npm start

step9: Check the output by using below url.

ex: http://localhost:3000

composing components using props

index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
```

```
const root=ReactDOM.createRoot(document.getElementById('root'));
```

```
root.render(
  <React.StrictMode>
    <App course="React"/>
  </React.StrictMode>
)
```

App.js

```
import Student from './Student';
```

```
function App(props)
{
  return (
    <Student crs={props.course}/>
  )
}
export default App;
```

Student.js

```
function Student(props)
{
    return (
        <h1>My Course Name : {props.crs}</h1>
    )
}
export default Student;
```

React CSS

CSS in React is used to style the React App or Component.

There are two ways available to add styling to your React App or Component with CSS.

- 1) Inline Styling
- 2) CSS Stylesheet

1)Inline CSS

Inline CSS represent by "style" attribute in React application.

The inline styles are specified with a JavaScript object in camelCase version of the style name.

ex:

App.js

```
import Student from "./Student";
function App()
{
    return <>
        <h1 style={{color:"green"}}>React Inline CSS</h1>
        <h1 style={{backgroundColor:"yellow"}}>React Inline CSS</h1>
        </>
    </>
    export default App;
```

The inline styling also allows us to create an object with styling information and refer it in the style attribute.

App.js

```
import Student from "./Student";
```

```
function App()
{
    const mystyle = {
        color: "white",
        backgroundColor: "DodgerBlue",
        padding: "10px",
        fontFamily: "Arial"
    };
    return <>
        <h1 style={mystyle}>React Inline CSS</h1>
        <h1 style={{backgroundColor:"yellow"}}>React Inline CSS</h1>
        </>
    </>
    export default App;
```

2) CSS Stylesheet

We can write styling in a separate file for your React application, and save the file with a .css extension.

Later we can import .css file in our required application.

ex:1

App.js

```
import Student from "./Student";
import './App.css';
function App()
{
    return <>
        <h1>React CSS styles</h1>
        <h1>React CSS styles</h1>
    </>
}
export default App;
```

App.css

```
body{
    background-color: yellow;
}
h1
{
    color:blue;
}
```

ex:2

App.js

```
import Student from "./Student";
import './App.css';
function App()
{
    return <>
        <h1 id="myId">React CSS styles</h1>
        <h1 className="myClass">React CSS styles</h1>
    </>
}
export default App;
```

App.css

```
body{
    background-color: yellow;
}
#myId
{
    color:blue;
}
.myClass
```

```
{  
  color:red;  
}
```

State

State is similar to props but it is a private and fully controlled by the component.

we can create a state only in class component but not in functional component.

It is possible to update the state or modify the state , where as props only for read only.

There are two ways to initialize the state in React component.

- 1)Directly inside class
- 2)Inside the Constructor

1. Directly inside class

```
class Student extends Component  
{  
  //define state  
  state={  
    name: "Anna Julie",  
    prop1: this.props.prop1  
  }  
  render()  
  {  
    -  
  }  
}
```

Note: The "state" property is referred as state.

"this" is a class instance property

example

Project structure

```
myapp6  
|  
|----node-modules  
|  
|----public  
|  |  
|  |----favicon.ico  
|  |----index.html  
|  |----manifest.json  
|  
|----src  
|  |  
|  |----index.js  
|  |----App.js  
|  
|----package.json  
|----README.md
```

step1: Develop React Application.

ex: E:/ReactProjects>npx create-react-app myapp6

step2: Open VSC editor from Reactprojects.

ex: E:/Reactprojects>code .

step3: Install "ES7 React " Plugin/Extension from Visual Studio Code
for shortcuts to create React Applications.

ex:

imr +tab

imrc + tab

imrd + tab

imp + tab

rcc - class component

rcfe - named function component

rafce - anonymous function component

conlg+ tab

step4: Add below code inside "App.js" file.

App.js

import React, { Component } from 'react'

```
export default class App extends Component {
    state={
        name:"Alan"
    }
    render() {
        return (
            <h1>Hello {this.state.name}</h1>
        )
    }
}
```

step5: move to myapp5

ex: E:/BUI-2pm/ReactProjects> cd myapp6

step6: Run the application.

ex: DE:/BUI-2pm/ReactProjects/myapp6>npm start

step7: Test the React Application.

ex: http://localhost:3000

ex:2

App.js

import React, { Component } from 'react'

export default class App extends Component {

```
    state={
        name:"Alan",
        roll:this.props.rollno
    }
```

render() {

```
        return (
            <div>
                <h1>Name: {this.state.name}</h1>
                <h1>RollNo: {this.state.roll}</h1>
            </div>
        )
    }
```

```
}
```

index.js

```
import App from './App';
import ReactDOM from 'react-dom/client';
import React from 'react';

const root=ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App rollno={501} />
  </React.StrictMode>
)
```

*Note: Here props property we are storing into a state.

2. Inside the Constructor

```
class App extends Component
{
  //constructor
  //props is optional
  constructor(props)
  {
    //it is required to call the parent class constructor
    super(props);

    //state
    this.state={
      name:"alan",
      prop1: this.props.prop1
    }
  }
  render()
  {
    -
  }
}
```

When the component class is created, The constructor is the first called so it is right place to add state.

The class instance has already been created in memory. So we can use "this" to set properties on it.

When we write a constructor ,make sure to call parent class constructor by using super(props) keyword.

When we call super with props ,React will make props available accross/access the component through this.props.

Project structure

myapp7

|

```

|-----node-modules
|
|-----public
|     |
|     |-----favicon.ico
|     |-----index.html
|     |-----manifest.json
|
|-----src
|     |
|     |-----index.js
|     |-----App.js
|
|-----package.json
|-----README.md

```

step1: Develop React Application.

ex: E:/ReactProjects>npx create-react-app myapp7

step2: Open VSC code editor.

ex: ReactProjects> code .

step3: Write below code inside "App.js" file in "src" folder (rcc).

Student.js

```
import React, { Component } from 'react'
```

```
export default class App extends Component {
```

```
    constructor()
```

```
{
```

```
    super();
```

```
    this.state={
```

```
        name: "Alan",
```

```
        roll: 101
```

```
}
```

```
}
```

```
render() {
```

```
    return (
```

```
        <div>
```

```
            <h1>Name: {this.state.name}</h1>
```

```
            <h1>RollNo: {this.state.roll}</h1>
```

```
        </div>
```

```
)
```

```
}
```

step4: move to myapp7

ex: E:/BUI-2pm/ReactProjects> cd myapp7

step5: Run the application.

ex: DE:/BUI-2pm/ReactProjects/myapp7>npm start

step6: Test the React Application.

ex: http://localhost:3000

ex:2

App.js

```
import React, { Component } from 'react'
export default class App extends Component {

    constructor(props)
    {
        super(props);
        this.state={

            name: "Alan",
            roll: this.props.rollno
        }
    }
    render() {
        return (
            <div>
                <h1>Name: {this.state.name}</h1>
                <h1>RollNo: {this.state.roll}</h1>
            </div>
        )
    }
}
```

index.js

```
import App from './App';
import ReactDOM from 'react-dom/client';
import React from 'react';

const root=ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <React.StrictMode>
        <App rollno={501} />
    </React.StrictMode>
)
```

Event Handling in React

Event: Action to which a javascript can respond is called event.

ex: clicking on button
 hovering of an element
 and etc.

Handling events on react Elements are same like handling events on DOM elements.

ex:

Javascript

```
<button onclick="f1()">clickMe</button>
```

React

```
<button onClick={handleClick}>clickMe</button> --> Function component
<button onClick={this.handleClick}>clickMe</button> --> Class component
```

Event Handling using Function component

Project structure

```
myapp8
|
|----node_modules
|
|----public
|      |
|      ---index.html
|      ---favicon.ico
|      ---manifest.json
|
|----src
|      |
|      ---index.js
|      ---index.css
|
|      |
|      ---App.js
|      ---App.css
|      ---App.test.js
|
|----package.json
|
```

step1: create a react project/application.

ex: ReactProjects>npx create-react-app myapp8

step2: Starts VSC code editor.

ex: ReactProjects> code .

step3: Move to the project.

ex: ReactProjects> cd myapp8

step4: Run the react application/project.

ex: ReactProjects/myapp8> npm start

ex:1

App.js

```
function App()
{
```

```
    function handleClick()
    {
        console.log("Button is clicked");
    }
    return (
        <button onClick={handleClick}>clickMe</button>
    )
}
export default App;
```

ex:2

App.js

```
function App()
{
    const handleClick=()=>
    {
        console.log("Button is clicked");
    }
    return (
        <button onClick={handleClick}>clickMe</button>
    )
}
export default App;
```

ex:3

```
import React from 'react'
```

```
function App() {

    function handleClick(e)
    {
        e.preventDefault();
        console.log("You have clicked");
    }

    return (
        <div>
            <a href="http://www.google.com" onClick={handleClick}> click </a>
        </div>
    )
}

export default App
```

Event Handling using class component

Project structure

```
myapp9
|
|---node_modules
|
|---public
    |
    |---index.html
    |---favicon.ico
    |---manifest.json
```

```

|
|---src
|     |
```

```
|---index.js  
|---index.css  
  
|  
|---App.js  
|---App.css  
|---App.test.js  
|  
|---package.json  
|
```

step1: create a react project/application.

ex: ReactProjects> create-react-app myapp9

step2: Starts VSC code editor.

ex: ReactProjects> code .

step3: Move to the project.

ex: ReactProjects> cd myapp9

step4: Run the react application/project.

ex: ReactProjects/myapp9> npm start

ex:1

App.js

```
import {Component} from "react";  
export default class App extends Component  
{  
  handleClick=()=>>  
  {  
    console.log("Button is clicked",this);  
  }  
  
  render()  
  {  
    return(  
      <button onClick={this.handleClick}>clickMe</button>  
    )  
  }  
}
```

index.js

```
import React from 'react';  
import ReactDOM from 'react-dom/client';  
import './index.css';  
import App from './App';  
import reportWebVitals from './reportWebVitals';  
  
const root = ReactDOM.createRoot(document.getElementById('root'));  
root.render(  
  <React.StrictMode>  
    <App />  
  </React.StrictMode>
```

```

);
// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();

```

update state

Using `setState()` method is used to update states.

ex:

```

this.state={
    name:"Alan"
}
this.setState({name:"Kelvin"});

```

ex:1

App.js

```

import {Component} from "react";
export default class App extends Component
{
    state={
        name : "Nancy",
        rollno: 101
    }

    handleClick=>
    {
        this.setState({name:"Lisa",rollno:501});
    }
    render()
    {
        return(
            <>
                <h1>Name : {this.state.name}</h1>
                <h1>RollNo : {this.state.rollno}</h1>
                <button onClick={this.handleClick}>Change state</button>
            </>
        )
    }
}

```

Interview Questions

Q) Difference between function component vs class component?

function component

It is also known as stateless component.

In a function component we will use `return` keyword.

It supports hooks.

Constructor is not used.

class component

It is a statefull component.

In a class component we will use `render()` method.

It does not support hooks.

Constructor is used.

Q) Difference between real dom vs virtual dom ?

Real dom

It updates slow.
Can directly update HTML.
Creates a new dom if element updates.
DOM manipulation is very expensive.
Too much of memory wastage.

virtual dom

It updates faster.
Can't directly update HTML.
Update the jsx if element updates.
DOM manipulation is very easy.
No memory wastage.

Q) Difference between props and state ?

props

Props are read-only.
Props are immutable.
Props allow us to pass data from one component to other components as an argument.
Props can be accessed by the child component.
Stateless component can have Props.

state

States are updatable.
State is mutable.
State holds information about the components.
State cannot be accessed by child components because it is private.
Statefull components can have state.

Phases of components in ReactJS

There are four Phases of components in ReactJS.

- 1)Mounting
- 2)Updating
- 3)Error Handling
- 4)Unmounting

1. Mounting

Mounting is a process of creating an element and inserting it in a DOM tree.

2. Updating

Updating is a process of changing state or props of a component and update changes to nodes already existing in the DOM.

3. Error Handling

Error Handling used when there is a error during rendering, in lifecycle method or in the constructor of any child component.

4. Unmounting

Unmounting is a process of removing elements from the DOM tree.
In general it will clear the reserved memory.

Q) Explain life cycle methods of mounting ?

Mounting phase contains four methods.

- 1) constructor()
- 2) getDerivedStateFromProps()
- 3) render()
- 4) componentDidMount()

Q) Explain life cycle methods of unmounting?

Unmounting phase contains one method.

1) componentWillUnmount()

Q) Explain life cycle methods of updating?

updating phase contains five methods.

- 1) getDerivedStateFromProps()
- 2) shouldComponentUpdate()
- 3) render()
- 4) getSnapshotBeforeUpdate()
- 5) ComponentDidUpdate()

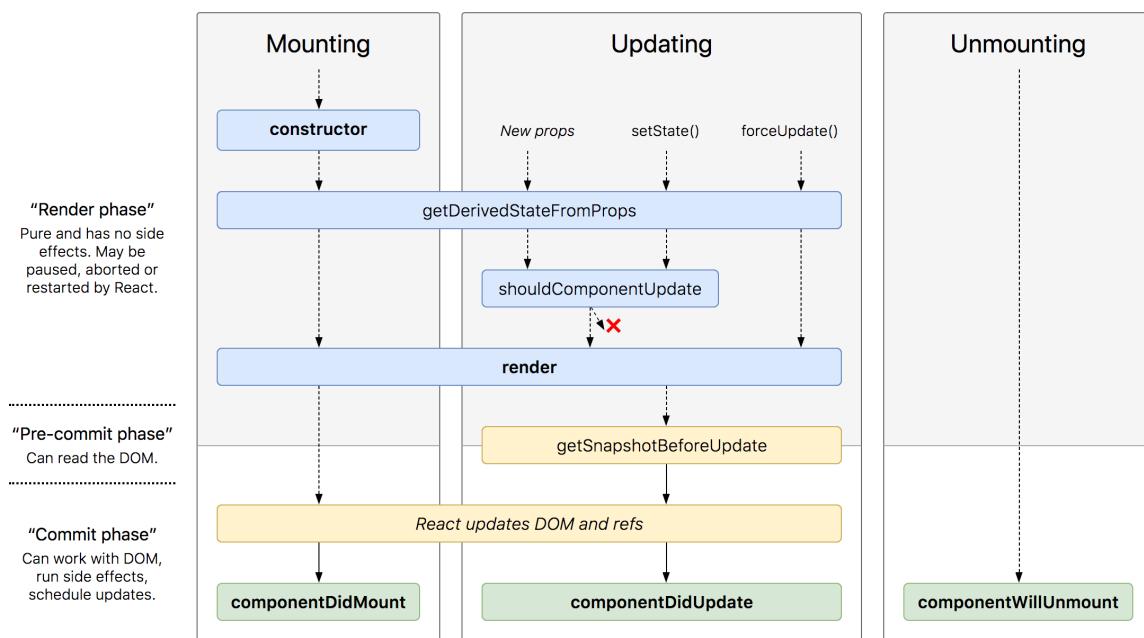


Diagram: react6.1

In react , all life cycle methods we can declare inside class component.

App.js

```
import React, { Component } from 'react'

export default class App extends Component {

  constructor()
  {
    console.log('constructor');
    super();

    this.state={
      name:"Alan"
    }
  }

  static getDerivedStateFromProps(props,state)
  {
    console.log('getDerivedStateFromProps')
```

```

}

render() {
  console.log('render');
  return (
    <>
    <h1>Name : {this.state.name}</h1>
    </>
  )
}

componentDidMount()
{
  console.log('componentDidMount')
}
}

```

Hooks

Hooks allow us to "hook" into React features such as state and lifecycle methods.

Hooks allow function components to have access to state , lifecycle methods and other React features.

Hooks allow us to use React without classes.It means you can use state and other React features without writing a class.

React provides a few built-In hooks like useState,useEffect and etc.

Hooks are new addition in React 16.8.

When use Hooks

If you write a function component and realize you need to add some state to it.

Rules of Hooks

There are 3 rules for hooks:

- 1)Hooks can only be called inside React function components.
- 2)Hooks can only be called at the top level of a component.
- 3)Hooks cannot be conditional

*Note: Hooks will not work in React class components.

Declaring State

A useState() is a Hook that allows us to add React state to function components.

We call it inside a function component to add some local state to it.

A useState() returns a pair - the current state value and a function that lets us update it.

React will preserve this state between re-renders.

We can call this function from an event handler or somewhere else.

Project structure

myapp10

|

|---node_modules

|

|---public

|

 |--favicon.ico

 |--index.html

```

|--manifest.json
|
|-----src
|     |
|     |---App.js
|     |
|     |---index.js
|
|-----package.json
|-----README.md

```

step1: create a react project.

ex: Reactprojects> npx create-react-app myapp10

step2: Open the VSC editor.

ex: Reactprojects> code .

step3: Move/Jump to myapp8 project.

ex: Reactprojects> cd myapp10

step4: Run the myapp8 project.

ex: Reactprojects/myapp10> npm start

step5: Test the application by using below request url.

ex: http://localhost:3000

App.js

```

import { useState } from "react";
function App()
{
    const [name,setName]=useState("Alan");

    const handleClick=()=>{
        {
            setName("Kelvin");
        }
        return (
            <div>
                <h1>Name : {name}</h1>
                <button onClick={handleClick}>clickMe</button>
            </div>
        )
    }
    export default App;
}

```

index.js

```

import Student from './Student';
import ReactDOM from 'react-dom/client';
import React from 'react';
import App from './App';

const root=ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <React.StrictMode>

```

```
<App />
</React.StrictMode>
)
```

Effect Hooks

The Effect Hook let us to perform side effects in function components.

Data fetching, setting up a subscription, and manually changing the DOM in React components are all examples of side effects.

useEffect()

A useEffect is a hook for encapsulating code that has "side effects".if we are familiar with React class life cycle methods. We can think of useEffect Hooks as componentDidMount, compnoentDidUpdate and componentWillUnmount combined.

useEffect =componentDidMount+ componentDidUpdate +componentWillUnmount

ex:

```
import React,{useEffect} from "react";
useEffect(Function)
```

or

```
useEffect(Function ,Array)
```

The function passes to useEffect will run after the render is committedto the screen.

Second argument to useEffect that is the array of values that the effect depends on.(It is for condition purpose).

*Note: We can call useEffect as many times we required.

ex:

```
useEffect(()=>
{
    console.log("Hello useeffect");
});
```

ex:

```
useEffect(()=>
{
    console.log("Hello useEffect");
},[count]);
```

What does useEffect do?

By using this Hook,we can tell react that your component needs to do something after render. React remember the function we passed and call it later after performing the DOM updates.

In this effect, we set the document title,we could also perform data fetching or call some other imperative API.

*Note: useEffect runs after the first render and after every update.

Project structure

```
myapp11
|
|---node_modules
|
|---public
|   |
```

```

|--favicon.ico
|--index.html
|--manifest.json
|
|-----src
|   |
|   ---App.js
|   |
|   ---index.js
|
|-----package.json
|-----README.md

```

step1: create a react project.

ex: Reactprojects> npx create-react-app myapp11

step2: Open the VSC editor.

ex: Reactprojects> code .

step3: Move/Jump to myapp9 project.

ex: Reactprojects> cd myapp11

step4: Run the myapp9 project.

ex: Reactprojects/myapp11> npm start

step5: Test the application by using below request url.

ex: http://localhost:3000

App.js

```

import { useState, useEffect } from "react";
function App()
{
  const [count, setCount] = useState(0);

  const handleClick = () =>
  {
    setCount(count + 1);
  }

  useEffect(() => {
    // Update the document title using the browser API
    document.title = `you have click for ${count} times`;
  });
  return (
    <div>
      <h1>You clicked {count} Times</h1>
      <button onClick={handleClick}>clickMe</button>
    </div>
  )
}
export default App;

```

index.js

```

import Student from './Student';

```

```

import ReactDOM from 'react-dom/client';
import React from 'react';
import App from './App';

const root=ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
)

```

React useContext Hook (Context API)

Context provides a way to pass the data through the component tree without passing props down manually at several level.

To do this without Context, we will need to pass the state(useState) as "props" through each nested component. This is called "props drilling".

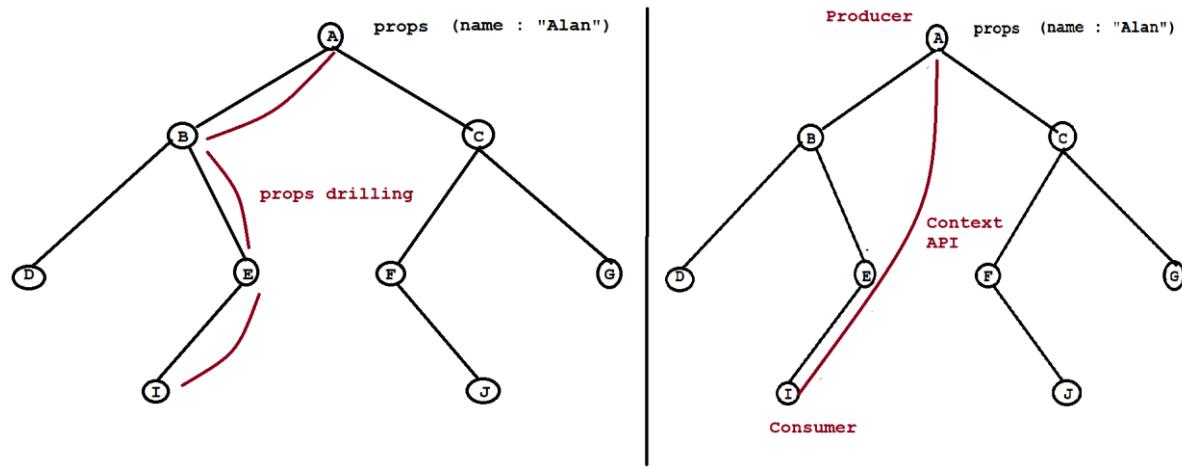


Diagram: react7.1

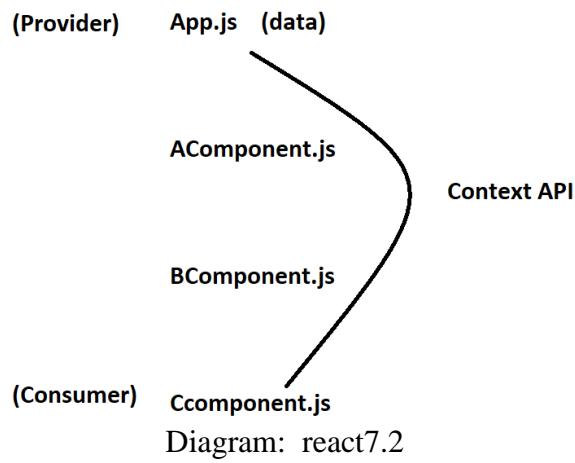
Project structure

```

myapp12
|
|-----node-modules
|
|-----public
|   |
|   |-----favicon.ico
|   |-----index.html
|   |-----manifest.json
|
|-----src
|   |
|   |-----index.js
|   |-----App.js
|   |-----Acomponent.js
|   |-----Bcomponent.js
|   |-----Ccomponent.js
|
|-----package.json

```

|-----README.md



step1: create a react project.

ex: Reactprojects> npx create-react-app myapp12

step2: Open the VSC editor.

ex: Reactprojects> code .

step3: Move/Jump to myapp10 project.

ex: Reactprojects> cd myapp12

step4: Run the myapp10 project.

ex: Reactprojects/myapp12> npm start

App.js

```
import React from 'react';
import Acomponent from "./Acomponent";
export const UseContext=React.createContext();
function App()
{
    return (
        <div>
            <UseContext.Provider value={'IHUB'}>
                <Acomponent/>
            </UseContext.Provider>
        </div>
    )
}
export default App;
```

Acomponent.js

```
import Bcomponent from "./Bcomponent";
function Acomponent()
{
    return (
        <Bcomponent/>
    )
}
export default Acomponent;
```

Bcomponent.js

```
import Ccomponent from "./Ccomponent";
function Bcomponent()
{
    return (
        <Ccomponent/>
    )
}
export default Bcomponent;
```

Ccomponent.js

```
import {UseContext} from "./App";
function Ccomponent()
{
    return (
        <div>
            <UseContext.Consumer>
                {
                    user => {
                        return <div>The value is : {user} </div>
                    }
                }
            </UseContext.Consumer>
        </div>
    )
}
export default Ccomponent;
```

index.js

```
import Student from './Student';
import ReactDOM from 'react-dom/client';
import React from 'react';
import App from './App';

const root=ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <React.StrictMode>
        <App />
    </React.StrictMode>
)
```

Custom Hooks

Hooks which are created by the user based on the application requirement are called custom hooks.

ex: myCustomHook()
 customHook()
 ihubHook()
 myCustomCounter()

Project Structure

```
myapp11
|
|---node_modules
|
|---public
|   |
|   |---favicon.ico
|   |---index.html
|   |---manifest.json
|
|---src
|   |
|   |---index.js
|   |---App.js
|   |---CustomHook.js
|
|---package.json
|---README.md
```

step1: Create a react project or application.

ex: Reactprojects> npx create-react-app myapp11

step2: Open VSC editor.

ex: Reactprojects> code .

step3: Move or Jump to myapp11 project.

ex: Reactprojects> cd myapp11

step4: Run the react application.

ex: Reactprojects/myapp11> npm start

step5: Test the react application.

ex: http://localhost:3000

step6: Create "CustomHook.js" file inside "src" folder.

ex:1

CustomHook.js

```
import React from 'react'
import {useState} from 'react'
function CustomHook()
{
    const [count,setCount]=useState(0);

    const handleClick=()=>{
        {
            setCount(count+1);
        }
        return(
            {
                count,
                handleClick
            }
        )
    }
}
```

```
        })
    }
export default CustomHook
```

App.js

```
import React from 'react'
import customHook from './CustomHook';
function App() {
  const data=customHook();

  return (
    <div>
      <h1>Count : {data.count}</h1>
      <button onClick={ data.handleClick }>Increment</button>
    </div>
  )
}
export default App
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

Images/Assets in ReactJS

We can set images/Asset in ReactJS using two ways.

1. Inside public Folder.
2. Inside src folder.

1. Inside public folder

If we put a file into a public folder, It will not be processed by webpack. Instead it will be copied into the build folder untouched.

To reference assets in the public folder, we need to use a special variable called PUBLIC_URL. Only files inside the public folder will be accessible by %PUBLIC_URL% prefix.

How to use image

1)
myapp13
|
|--public
|
|---rock.jpg

index.html

```

```

2)
myapp
|
|--public
|
|---images
|
|---rock.jpg

index.html

```

```

If we want to use Image in Javascript file.

App.js

```
<img src={process.env.PUBLIC_URL + "/rock.jpg"} />  
<img src={process.env.PUBLIC_URL + "/images/rock.jpg"} />
```

ex:1

index.html

```
-  
-  
-  
  
<div id="root"></div>  
  
-  
-  
-
```

*Note: Mostly of the time we are displaying images in Component only.

ex:

App.js

```
import React, { Component } from 'react'
```

```
export default class App extends Component {  
  render() {  
    return (  
      <div>  
        <img src={process.env.PUBLIC_URL+"team1.jpeg"} alt="mypic"/>  
      </div>  
    )  
  }  
}
```

```
}
```

```
index.js
import React from 'react';
import ReactDOM from 'react-dom';
import App from "./App";

//render the component to index.html
ReactDOM.render(<App />,document.getElementById("root"));
```

2) Inside src folder

we can import a file right in a Javascript module. This tell webpack to include that file in the bundle.

How to use

```
1)
myapp
|
|---src
    |
    |---rock.jpg
```

```
App.js
import pic from "./rock.jpg";
<img src={pic} alt="mypic" />
```

This ensures that when the project is built. Webpack wil correctly movethe images into the build folder and provide us with correct paths.

ex:

```
App.js
import React, { Component } from 'react'
import pic from "./team1.jpeg";

export default class App extends Component {
  render() {
    return (
      <div>
        <img src={pic} alt="mypic"></img>
      </div>
    )
  }
}
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from "./App";

//render the component to index.html
ReactDOM.render(<App />,document.getElementById("root"));
```

React Router

Routing is a process in which a user is directed to different pages based on their actions or requests.

ReactJS Router is mainly used for developing Single Page Web Applications.

React Router is used to define multiple routes in the application.

When a user types a specific URL into the browser, and if this URL path matches any 'route' inside the router file, the user will be redirected to that particular route.

React Router is a standard library system built on top of the React and used to create routing in the React application using React Router Package.

React contains three different packages for routing.

1) react-router:

It provides the core routing components and functions for the React Router applications.

2) react-router-native:

It is used for mobile applications.

3) react-router-dom:

It is used for web applications design.

*Note:

It is not possible to install react-router directly in your application.

To use react routing, first, you need to install react-router-dom modules in your application.

We have two types of router components.

1. <BrowserRouter>:

It is used for handling the dynamic URL.

2. <HashRouter>:

It is used for handling the static request.

Project structure

myapp12

|

|-----node-modules

|

|-----public

|

|-----favicon.ico

|-----index.html

|-----manifest.json

|

|-----src

|

|-----index.js

|-----App.js (Routing File)

|-----Home.js

|-----About.js

|-----Contact.js

|-----Error.js

|

|-----package.json

|-----README.md

step1: create react "myapp12" project in VSC.
ex: projects>npx create-react-app myapp12
step2: Move to myapp12 project.
ex: project>cd myapp12
step3: install react router dom.
ex: project/myapp12>npm install --save react-router-dom
step4: Restart the application .
ex: myapp14> npm start
step5: create App.js,Home.js,About.js ,Contact.js and Error.js component inside "src" folder.

App.js

```
import Home from './Home';
import Contact from './Contact';
import About from './About';
import Error from './Error'
import { BrowserRouter, Routes, Route } from "react-router-dom";

function App() {
  return (
    <div>
      <BrowserRouter>
        <Routes>
          <Route exact path="/" element={<Home />}/>
          <Route path="/about" element={<About />}/>
          <Route path="/contact" element={<Contact />}/>
          <Route path="*" element={<Error />}/>
        </Routes>
      </BrowserRouter>
    </div>
  );
}
export default App;
```

Home.js

```
function Home()
{
  return (
    <div>
      <h1>Home</h1>
    </div>
  )
}
export default Home;
```

About.js

```
function About()
{
  return (
    <div>
      <h1>About</h1>
    </div>
  )
}
export default About;
```

```
        </div>
    )
}

export default About;
```

Contact.js

```
function Contact()
{
    return (
        <div>
            <h1>Contact</h1>
        </div>
    )
}

export default Contact;
```

Error.js

```
function Error()
{
    return(
        <div>
            <h1>OOPS! 404 Error </h1>
        </div>
    )
}

export default Error;
```

step6: create index.js component to render the output inside "src" folder.

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <React.StrictMode>

        <App/>
    </React.StrictMode>
);
```

step7: Test the application by using below url's.

ex: http://localhost:3000/
http://localhost:3000/home
http://localhost:3000/about
http://localhost:3000/contact
http://localhost:3000/gallery
http://localhost:3000/services

Adding Navigation using Link component

A Link component is used to create links which allow to navigate on different URLs and render its content without reloading the webpage.

ex:2

App.js

```
import Home from './Home';
import Contact from './Contact';
import About from './About';
import Error from './Error'

import {Link, Routes, Route, BrowserRouter } from 'react-router-dom'
function App() {
  return (
    <div>
      <BrowserRouter>

        <nav >
          <Link style={{display:"block"}} to="/">Home</Link>
          <Link style={{display:"block"}} to="/about">About Us</Link>
          <Link style={{display:"block"}} to="/contact">Contact US</Link>
        </nav>
        <Routes>
          <Route exact path="/" element={<Home />}/>
          <Route path="/about" element={<About />}/>
          <Route path="/contact" element={<Contact />}/>
          <Route path="*" element={<Error />}/>
        </Routes>
      </BrowserRouter>
    </div>
  );
}
export default App;
```

Home.js

```
function Home()
{
  return (
    <div>
      <h1>Home</h1>
    </div>
  )
}
export default Home;
```

About.js

```
function About()
{
  return (
    <div>
```

```
        <h1>About</h1>
      </div>
    )
}
export default About;
```

Contact.js

```
function Contact()
{
  return (
    <div>
      <h1>Contact</h1>
    </div>
  )
}
export default Contact;
```

Error.js

```
function Error()
{
  return(
    <div>
      <h1>OOPS! 404 Error </h1>
    </div>
  )
}
export default Error;
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>

    <App/>

  </React.StrictMode>
);
```

Bootstrap in React

A Single-page applications gaining popularity over the last few years, so many front-end frameworks have introduced such as Angular, Vue, Ember, etc. As a result, jQuery is not a necessary requirement for building web apps.

Currently, React is mostly used JavaScript library for building web applications, and Bootstrap become the most popular CSS framework.

Let see how to use bootstrap in react applications.

Project structure

```
myapp14
|
|----node_modules
|
|----public
|    |
|    |---favicon.ico
|    |---index.html
|    |---manifest.json
|
|----src
|    |
|    |---index.js
|    |
|    |---App.js
|
|----package.json
|
|----README.md
```

step1: create a react project i.e myapp14.

ex: Reactprojects> npx create-react-app myapp14

step2: Open the VSC code editor.

ex: Reactprojects> code .

step3: Move/Switch to myapp14 project.

ex: Reactprojects> cd myapp14

step4:

Install Bootstrap package.

ex: Reactprojects/myapp13> npm install bootstrap

step5: Run the react application.

ex: Reactprojecs/myapp14> npm start

step6: Create a App.js file inside "src" folder.

App.js

```
function App()
{
  return(
    <div className="container mt-5">
      <button className="btn btn-outline-primary">clickMe</button>
    </div>
  )
}
export default App;
```

step7: Import bootstrap package inside "index.js" file.

index.js

```
import React from 'react';
```

```

import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import './node_modules/bootstrap/dist/css/bootstrap.css';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

reportWebVitals();

```

step8: Test the application by using below request url.

ex: http://localhost:3000

React Forms

Forms are an integral part of any modern web application.

It allows the users to interact with the application as well as gather information from the users.

Forms can perform many tasks that depend on the nature of your business requirements and logic such as authentication of the user, adding user, searching, filtering, booking, ordering, etc.

A form can contain text fields, buttons, checkbox, radio button, etc.

Creating Form

React offers a stateful, reactive approach to build a form.

The component rather than the DOM usually handles the React form.

In React, the form is usually implemented by using controlled components.

Controlled component

In the controlled component, the input form element is handled by the component rather than the DOM. Here, the mutable state is kept in the state property and will be updated only with `setState()` method.

Controlled components have functions that govern the data passing into them on every `onChange` event, rather than grabbing the data only once, e.g., when you click a submit button. This data is then saved to state and updated with `setState()` method. This makes component have better control over the form elements and data.

Project structure

```

myapp15
|
|---node_modules
|
|---public
    |
    |---favicon.ico

```

```

|---index.html
|---manifest.json

|-----src
|       |
|       |---index.js
|       |
|       |---App.js

|
|-----package.json
|
|-----README.md

```

- step1: create a react project i.e myapp15.
 ex: Reactprojects> npx create-react-app myapp15
- step2: Open the VSC code Editor.
 ex: Reactprojects> code .
- step3: Switch/Move to myapp15 project.
 ex: Reactprojects> cd myapp15
- step4: Install bootstrap package.
 ex: Reactprojects/myapp14> npm install bootstrap
- step5: Run the react application.
 ex: Reactprojects/myapp14> npm start
- step6: Import Bootstrap package inside "index.js" file.

index.js

```

import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import './node_modules/bootstrap/dist/css/bootstrap.css';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();

```

- step7: Create App.js file inside "src" folder.

App.js

```

import {useState} from 'react';

```

```
function App()
{
  const [userRegistration, setUserRegistration]=useState({
    username:"",
    password:"",
    date:"",
    category:""
  })

  const handleClick=(e)=>
  {
    const name=e.target.name;
    const value=e.target.value;
    //set to state
    setUserRegistration({ ... userRegistration,[name]:value });
  }

  const handleSubmit=(e)=>
  {
    e.preventDefault();
    setUserRegistration({username:"",password:"",date:"",category:""});
  }

  return(
<div className="container mt-4">

  <form onSubmit={handleSubmit}>
  <div className="row w-50">
  <h1 className="text-center" ><u>React Form </u></h1>
  <label htmlFor="username" className="my-3">UserName:</label>
  <input type="text" name="username" autocomplete="off"
    className="form-control"
    value={userRegistration.username}
    onChange={handleClick}/>

  <label htmlFor="password" className="my-3">Password:</label>
  <input type="password" name="password" autocomplete="off"
    className="form-control"
    value={userRegistration.password}
    onChange={handleClick}/>

  <label htmlFor="date" className="my-3">Date:</label>
  <input type="date" name="date" autocomplete="off"
    className="form-control"
    value={userRegistration.date}
    onChange={handleClick}/>

  <label htmlFor="category" className="my-3">Category</label>

```

```

        <select name="category" className="form-control"
            value={userRegistration.category}
            onChange={handleClick}>
            <option value="">none</option>
            <option value="entertainment">Entertainment</option>
            <option value="drama">Drama</option>
            <option value="action">Action</option>
        </select>

        <button className="btn btn-primary mt-4 w-100"> submit </button>
    </div>

    </form>
</div>

)

}

export default App;

```

step8: Test the application by using below request url.
ex: http://localhost:3000

Lists in ReactJs

Lists are used to display data in an ordered format and mainly used to display menus on websites. In React, Lists can be created in a similar way as we create lists in JavaScript. Let us see how we transform Lists in regular JavaScript.

The map() function is used for traversing the lists.

ex:

```

Project structure
myapp16
|
|---node_modules
|
|---public
|   |
|   |---favicon.ico
|   |---index.html
|   |---manifest.json
|
|---src
|   |
|   |---index.js
|   |---App.js
|
|---package.json
|---README.md

```

step1: create a react project i.e myapp16.
ex: Reactprojects> npx create-react-app myapp16

step2: Open the VSC code editor.

ex:

Reactprojects> code .

step3: Move/Switch to myapp16 project.

ex: Reactprojects> cd myapp16

step4: Run the react application.

ex; Reactprojects/myapp16> npm start

step5: Create App.js file inside "src" folder.

App.js

```
import React, { Component } from 'react'
export default class App extends Component {
  render() {

    var arr=[10,20,30,40];

    var newArr=arr.map((element)=>
    {
      return <li>{element}</li>
    })

    return (
      <ul>
        {newArr}
      </ul>
    )
  }
}
```

step6: Test the application by using below request url.

ex: http://localhost:3000

ex:2

App.js

```
import React, { Component } from 'react'
export default class App extends Component {
  state={
    users:[
      {pid:101,pname:"LG",pprice:10000},
      {pid:102,pname:"LAVA",pprice:20000},
      {pid:103,pname:"MI",pprice:30000},
      {pid:104,pname:"SAMSUNG",pprice:40000}
    ]
  }
}
```

render() {

```
  var newArr=this.state.users.map(user=>
  {
    return <h1>Id: {user.pid} Name: {user.pname} Price: {user.pprice}</h1>
```

```

        })
      return (
        <div>
          {newArr}
        </div>
      )
    }
  }
}

```

ex:3

App.js

```

import React, { Component } from 'react'
export default class App extends Component {
  state={
    users:[
      {pid:101,pname:"LG",pprice:10000},
      {pid:102,pname:"LAVA",pprice:20000},
      {pid:103,pname:"MI",pprice:30000},
      {pid:104,pname:"SAMSUNG",pprice:40000}
    ]
  }
}

```

render() {

```

    var newArr=this.state.users.map(user=>
    {
      return <tr><td>{user.pid}</td> <td> {user.pname}</td> <td>{user.pprice}</td></tr>
    })
  return (
    <div>
      <table border={1} width="100%">
        <thead>
          <tr>
            <th>ID</th>
            <th>NAME</th>
            <th>PRICE</th>
          </tr>
        </thead>
        <tbody>
          {newArr}
        </tbody>
      </table>
    </div>
  )
}

```

Key in ReactJS

A key is a special string attribute you need to include when creating lists of elements. Keys help react identify which items have changed are added or are removed.

ex:

App.js

```
import React, { Component } from 'react'
export default class App extends Component {
  state={
    users:[
      {pid:101,pname:"LG",pprice:10000},
      {pid:102,pname:"LAVA",pprice:20000},
      {pid:103,pname:"MI",pprice:30000},
      {pid:104,pname:"SAMSUNG",pprice:40000}
    ]
  }
  render() {
    var newArr=this.state.users.map(user=>
    {
      return <tr key={user.pid}><td>{user.pid}</td> <td> {user.pname}</td>
<td>{user.pprice}</td></tr>
    })
    return (
      <table border={1} width="100%">
        <thead>
          <tr>
            <th>ID</th>
            <th>NAME</th>
            <th>PRICE</th>
          </tr>
        </thead>
        <tbody>
          {newArr}
        </tbody>
      </table>
    )
  }
}
```

Axios

Axios is used to make HTTP request (GET,POST,PUT,DELETE).

Using axios we can give the request to Rest API's.

We can install axios by using below command.

ex: reactprojects> npm install axios

or

reactprojects> yarn add axios

Project structure

```
myapp17
|
|---node_modules
|
|---public
|   |
|   |---favicon.ico
|   |---index.html
|   |---manifest.json
|
|---src
|   |
|   |---index.js
|   |---App.js
|   |---FetchApi.js
|
|---package.json
|---README.md
```

step1: create a react project i.e myapp17.

ex: Reactprojects> npx create-react-app myapp17

step2: Open the VSC code editor.

ex: Reactprojects> code .

step3: Move/Switch to myapp17 project.

ex: Reactprojects> cd myapp17

step4: Install axios in myapp17 project.

ex: Reactprojects/myapp17> npm install axios

step5: Run the react application.

ex: Reactprojects/myapp17> npm start

step6: Create App.js file inside "src" folder.

App.js

```
import FetchApi from './FetchApi';
function App()
{
  return (
    <FetchApi/>
  )
}
export default App;
```

step7: Arange one REST API for fetching the data.

ex: https://jsonplaceholder.typicode.com/users

step8: Create FetchApi.js file inside "src" folder.

FetchApi.js

```
import {useState} from 'react';
import axios from 'axios';
function FetchApi()
```

```

{
  const [data,setData]=useState([])
  const handleClick=()=>
  {
    axios.get("https://jsonplaceholder.typicode.com/users")
    .then(response=>
    {
      setData(response.data)
    })
    .catch(error=>
    {
      this.setData(error);
    })
  }
  return (
    <div>
      <center>
        <button onClick={handleClick}>Fetch API </button>
      </center>
      <table border={1} width="100%">
        <thead>
          <tr>
            <th>ID</th>
            <th>NAME</th>
            <th>USERNAME</th>
            <th>EMAIL</th>
          </tr>
        </thead>
        <tbody>
          {
            data.map(data=>
            {
              return <tr>
                <td>{data.id}</td>
                <td>{ data.name }</td>
                <td>{ data.username }</td>
                <td>{ data.email }</td>
              </tr>
            ))
          }
        </tbody>
      </table>
    </div>
  )
}
export default FetchApi;

```

step9: Test the application by using below request url.
 ex: http://localhost:3000