# Table of contents



# Java Introduction

| | |
|---|---|
| Author | James Gosling |
| Vendor | Sun Micro System(which has since merged into Oracle Corporation) |
| Project name | Green Project |
| Type | open source & free software |
| Initial Name | OAK language |
| Present Name | java Extensions : .java & .class & .jar |
| Initial version | jdk 1.0 (java development kit) |
| Present version | java 8 2014 |
| Operating System | multi Operating System |
| Implementation Lang | c, cpp…… |
| Symbol | coffee cup with saucer |
| Objective | To develop web applications |
| SUN | Stanford Universally Network |
| Slogan/Motto | WORA(write once run anywhere) |

## Importance of Core Java:-

According to the SUN 3 billion devices run on the java language only.

1) Java is used to develop Desktop Applications such as MediaPlayer,Antivirus etc.

2) Java is used to develop Web Applications such as irctc.co.in etc.

3) Java is used to develop Enterprise Application such as Banking applications like Finacle.

4) Java is used to develop Mobile Applications.

5) Java is used to develop Embedded System.

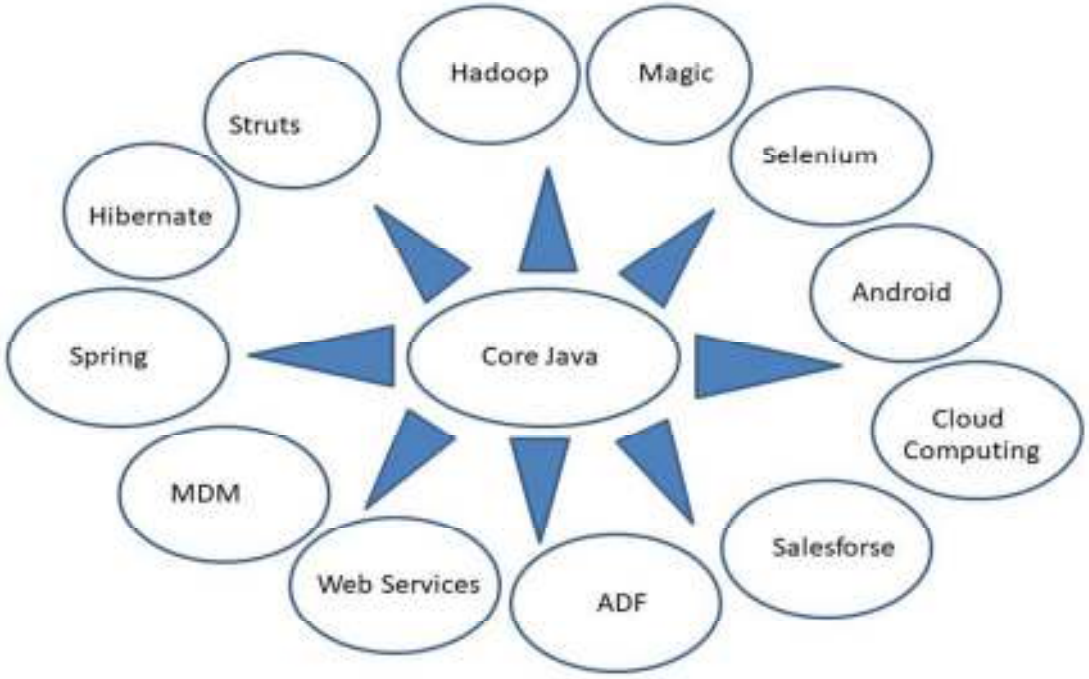
6) Java is used to develop SmartCards.

7) Java is used to develop Robotics.

8) Java is used to develop Games ……..etc.

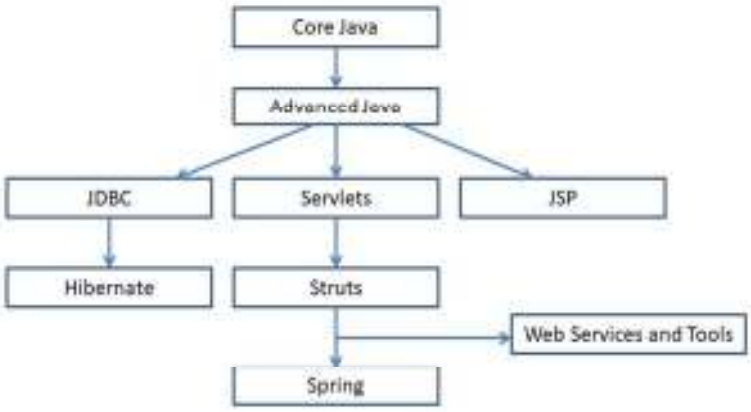
 

## Technologies depend on Core Java:-




## Learning Process:-

The following figure describe how to learn Java Technology.



## Java Keywords:-

There are 50 keywords in java. true, false and null seems like keywords but there are literals. Usage wise lists of java keywords are given below.

| **Data Types** | **Flow Controls** | **Method Level** | **Exception Handling** | **Modifiers** |
|---|---|---|---|---|
| Byte | If | void | try | public |
| Short | else | return | catch | private |
| Int | switch | | finally | protected |
| Long | case | **Object Level** | throw | static |
| Float | break | new | throws | final |
| Double | default | this | | abstract |
| Char | For | super | **Source File** | native |
| Boolean | while | instanceof | class | transient |
| | Do | | extends | volatile |
| | continue | **1.5 Version** | interface | synchronized |
| | | enum | implements | strictfp |
| | | assert | package | |
| | | | import | |
| | | **Unused** | | |
| | | const | | |
| | | goto | | |

## JAVA Features:-

1. Simple
2. Object Oriented
3. Platform Independent
4. Architectural Neutral
5. Portable
6. Robust
7. Secure
8. Dynamic
9. Distributed
10. Multithreaded
11. Interpretive
12. High Performance

## 1. Simple:-

Java is a simple programming language because:

- Java technology has eliminated all the difficult and confusion oriented concepts like pointers, multiple inheritance in the java language.
- The c, cpp syntaxes easy to understand and easy to write. Java maintains C and CPP syntax mainly hence java is simple language.
- Java tech takes less time to compile and execute the program.

## 2. Object Oriented:-

Java is object oriented technology because to represent total data in the form of object. By using object reference we are calling all the methods, variables which is present in that class.

## 3. Platform Independent:-

- Compile the Java program on one OS (operating system) that compiled file can execute in any OS (operating system) is called Platform Independent Nature.
- The java is platform independent language. The java applications allow its applications compilation one operating system that compiled (.class) files can be executed in any operating system.

## 4. Architectural Neutral:-

Java tech applications compiled in one Architecture (hardware----RAM, Hard Disk) and that Compiled program runs on any hardware architecture (hardware) is called Architectural Neutral.

## 5. Portable:-

In Java tech the applications are compiled and executed in any OS(operating system) and any Architecture(hardware) hence we can say java is a portable language.

## 6. Robust:-

Any technology if it is good at two main areas it is said to be ROBUST

1. Exception Handling
2. Memory Allocation

JAVA is Robust because

- JAVA is having very good predefined Exception Handling mechanism whenever we are getting exception we are having meaning full information.
- JAVA is having very good memory management system that is Dynamic Memory (at runtime the memory is allocated) Allocation which allocates and deallocates memory for objects at runtime.

## 7. Secure:-

- To provide implicit security Java provide one component inside JVM called Security Manager.
- To provide explicit security for the Java applications we are having very good predefined library in the form of java.Security.package.

## 8. Dynamic:-

Java is dynamic technology it follows dynamic memory allocation(at runtime the memory is allocated) and dynamic loading to perform the operations.

## 9. Distributed:-

By using JAVA technology we are preparing standalone applications and Distributed applications.

- Standalone applications are java applications it doesn't need client server architecture.
- web applications are java applications it need client server architecture.
- Distributed applications are the applications the project code is distributed in multiple number of jvm's.

## 10. Multithreaded:-

- Thread is a light weight process and a small task in large program.
- If any tech allows executing single thread at a time such type of technologies is called single threaded technology.
- If any technology allows creating and executing more than one thread called as multithreaded technology called JAVA.

## 11. Interpretive:-

JAVA tech is both Interpretive and Completive by using Interpretator we are converting source code into byte code and the interpretator is a part of JVM.

## 12. High Performance:-

If any technology having features like Robust, Security, Platform Independent, Dynamic and so on then that technology is high performance.

## Install the software and set the path:-

1) Download the software.

2) Install the software in your machine.

3) Set the environmental variable.

Download the software from internet based on your operating system. The software is different from 32-bit operating and 64-bit operating system.

To download the software open the fallowing web site.
http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

for 32-bit operating system please click on

Windows x86 :- 32- bit operating system

for 64-bit operating system please click on

Windows x64 :- 64-bit operating system

After installing the software the java folder is available in the fallowing location

Local Disk c: ----→program Files------→java--→jdk(java development kit),jre(java runtime environment) To check whether the java is installed in your system or not go to the command prompt. To open the command prompt Start --------→run----→open: cmd---→ok Command prompt is opened.

In the command prompt type :- javac

'javac' is not recognized is an internal or external command, operable program or batch file

Whenever we are getting above information at that moment the java is installed but the java is not working properly.

Whenever we are typing javac command on the command prompt

1) Operating system will pickup javac command search it in the internal operating system calls. The javac not available in the internal command list .

2) Then operating system goes to environmental variables and check is there any path is sets or not. up to now we are not setting any path. So operating system don't know anything about javac command Because of this reason we are getting error message.

**Hence we have to environmental variables. The main aim of the setting environmental variable is to make available the fallowing commands javac,java,javap (softwares) to the operating system.**

## To set the environmental variable:-

My Computer (right click on that) ---->properties----->Advanced--->Environment Variables----> User variables→new--→variable name : Path

Variable value: C:\programfiles\java\jdk1.6.0_11\bin; -----→ok----→ok

Now the java is working good in your system. open the command prompt to check once C:>javac---------now list of commands will be displayed .

**Steps to Design a First Application:-**

**Step-1:- Select Editor.**

**Step-2:- Write the application.**

**Step-3:- save the application.**

**Step-4:- Compilation Process.**

**Step-5:- Execution process.**

## Step1:- Select Editor

Editor is a tool or software it will provide very good environment to develop java application.
Ex :- Notepad, Notepad++,edit Plus…..etc

**IDE ( Integrated development Environment ):-** IDE is providing very good environment to develop the application and it is real-time standard but don't use IDE to develop core java applications.

**Editor vs. IDE:-** If we are using IDE to develop core java application then 75% work is done by IDE like

1) Automatic compilation.

2) Automatic import.

3) It shows all the methods of classes.

4) Automatically generate try catch blocks and throws (Exception handling)

5) It is showing the information about how to fix the bug……etc

And remaining 25% work is down by developer

If we are using EditPlus software to develop application then 100% work done by user only.

## Step 2:- Write a program.

- Write the java program based on the java API(Application Programming Interface) rule and regulations .
  **Open editplus --->file ----->new ------>click on java (it display sample java application )**
- Java is a case Sensitive Language so while writing the program you must take care about the case (Alphabet symbols).

**Example application:-**

```
import java.lang.System;
import java.lang.String;
 class Test //class declaration
{
 //class starts
 public static void main(String[] args) //program starting point
 {
 //main starts
 System.out.println("Hello World");  //printing statement
} //main ends
} //class ends
```

## Step3:- save the application.

- After writing the application must save the application by using (.java) extension.
- While saving the application must fallow two rules :-

i.) If the source file contains public class then public class and the name and Source file must be same (publicClassName.java). Otherwise compiler generate error message.

ii.) if the source file does not contain public class then save the source file with any name(anyName.java) .

## Step-4:- Compilation process.

Compile the java application by using **javac** command.

Syntax:-

C:\>javac Test.java

## Step-5:- Execution process.

Run /execute the java application by using **java** command.

## Syntax:-

java class-name

**java Test**

## Example application :-

- The default package in java is java.lang package it means if we are importing or not by default that package is imported.
- In below example importing classes are optional.

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

## Example application:-

The class contains main method is called **Main class** and java allows to declare multiple main class in a single source file.

```
class Test1
{
public static void main(String[] args)
{
 System.out.println("Test1 World!");
}
 }
class Test2
{
 public static void main(String[] args)
{
System.out.println("Test2 World!");
}
}
class Test3
{
public static void main(String[] args)
{
System.out.println("Test3 World!");
}
}
```

C:\>java Test1

**Test1 World**

C:\>java Test2

**Test2 World**

C:\>java Test3

**Test3 World**

## Java coding conventions:-

## Classes:-

- Class name start with upper case letter and every inner word starts with upper case letter.
- This convention is also known as **camel** case convention.
- The class name should be nouns.

Ex:- **S**tring, **S**tring**B**uffer, **I**nput**S**tream**R**eader ……etc.

## Interfaces:-

- Interface name starts with upper case and every inner word starts with upper case letter.
- This convention is also known as **camel** case convention.
- The Interface name should be nouns.

Ex:- **S**erializable, **C**loneable, **R**andom**A**ccess…….etc

## Methods :-

- Method name starts with lower case letter and every inner word starts with upper case letter.
- This convention is also known as mixed case convention
- Method name should be verbs.

Ex:- post(), char**A**t(), to**U**pper**C**ase(), compare**T**o**I**gnoreCase()…….etc.

## Variables:-

- Variable name starts with lower case letter and every inner word starts with upper case letter.
- This convention is also known as mixed case convention.

Ex :- out, in, page**C**ontext ……etc.

## Package:-

- Package name is always must written in lower case letters

Ex:- java.long, java.io, java.util …….etc.

## Constants:-

- While declaring constants all the words are uppercase letters .

Ex: MAX_PRIORITY MIN_PRIORITY NORM_PRIORITY

**NOTE:-The coding standards are applicable for predefined library not for user defined library .But it is recommended to fallow the coding standards for user defined library also.**

**Java Identifiers:-** Any name in the java program like variable name, classname, methodname, interface name is called identifier.

```
class Test
{
void add()
{
int a=10;
int b=20;
}
}
```

**Test---→Identifier**
**add----→Identifier**
**a-------→Identifier**
**b------→ Identifier**

## Rules to declare identifiers:-

1. the java identifiers should not start with numbers, it may start with alphabet symbol and underscore symbol and dollar symbol.

a. int abc=10;---→valid

b. int 2abc=20;--→not valid

c. int _abc=30;--→valid

d. int $abc=40;--→valid

e. int @abc=50;-→not valid

2. The identifier will not contains symbols like + , - , . , @ , # , *…….

3. The identifier should not duplicated.

4. In the java applications it is possible to declare all the predefined class names and predefined interfaces names as a identifier. But it is not recommended to use.

```
class Test
{
public static void main(String[] args)
{
int String=10;  //predefind String class
int Serializable=20; //predified Seriaiable Interface
```

```
float Exception=10.2f; //predefined Exception class
System.out.println(String);
System.out.println(Serializable);
System.out.println(Exception);
}
}
```

## Java.util.Scanner(Dynamic Input):-

1. **Scanner** class present in **java.util** package and it is introduced in 1.5 version.

2. **Scanner** class is used to take dynamic input from the keyboard.

**Scanner s = new Scanner(System.in);**

**to get int value ----> s.nextInt()**

**to get float value ---> s.nextFloat()**

**to get byte value ---> s.nextbyte()**

**to get String value ---> s.next()**

**to get single line ---> s.nextLine()**

**to close the input stream ---> s.close()**

## Example Application:-

```
import java.util.*;
class Test
{
public static void main(String[] args)
{
Scanner s=new Scanner(System.in); //used to take dynamic input from keyboard
System.out.println("enter emp hobbies");
String ehobbies = s.nextLine();
System.out.println("enter emp no");
 int eno=s.nextInt();
 System.out.println("enter emp name");
String ename=s.next();
System.out.println("enter emp salary");
float esal=s.nextFloat();
System.out.println("*****emp details*****");
```

```
System.out.println("emp no----->"+eno);
System.out.println("emp name---->"+ename);
 System.out.println("emp sal------>"+esal);
System.out.println("emp hobbies------>"+ehobbies);
s.close(); //used to close the stream
}
}
```

## WAP in java to find volume and surface area of cuboid

```
import java.util.Scanner;
class Cuboid
{
public static void main(String [] args)
{
int l,b,h;
Scanner sc=new Scanner(System.in);
System.out.print(“Enter length of cuboid : ”);
l=sc.nextInt();
System.out.print(“Enter breadth of cuboid : ”);
b=sc.nextInt();
System.out.print(“Enter height of cuboid : ”);
h=sc.nextInt();
int v=l*b*h;
int sa=2*(l*b+b*h+h*l);
System.out.println(“Volume of cuboid : ”+v);
System.out.println(“Surface Area of cuboid : ”+sa);
}
}
```

## Technical Tasks

1. Develop a program in Java to find area and perimeter of circle.
2. Develop a program in java to find simple interest and compound interest.
3. Develop a program in java to make a simple calculator.
4. Develop a program in java to swap two numbers without using third variable.
5. Develop a program in java to accept a coordinate point in a XY coordinate system and determine its quadrant.
6. Accept the salary of an employee from the user. Calculate the gross salary on the following basis:

| BASIC | HRA | DA |
|---|---|---|
| 1 – 4000 | 10% | 50% |
| 4001 – 8000 | 20% | 60% |
| 8001 - 12000 | 25% | 70% |
| 12000 and above | 30% | 80% |

# Flow Controls in Java

**Decision Controls:-** Decision Controls are used for decision making. The decision controls in java are given below:-

1. If statement
2. If – else statement
3. Nested if – else statement
4. Ladder if – else statement
5. Switch

**if Statement:-** if is a keyword which works like decision control. We attach a condition with if statement. If given condition is true then code will executed and if given condition is false then it do nothing.

## Syntax:-

```
if(Condition)
{
//Code
}
```

## Example Application:-

```
import java.util.Scanner;
class DecisionControlDemo1
{
public static void main(String [] args)
{
int num;
Scanner sc=new Scanner(System.in);
System.out.print("Enter a number : ");
num=sc.nextInt();
If(num==1)
{
System.out.println("Hi……");
}
System.out.println("Outside of if statement");
}
}
```

**O/P 1:-**

```
Enter a number : 1
Hi…….
Outside of if statement
```

**O/P 2:-**

```
Enter a number : 2
Outside of if statement
```

**if-else Statement:-** if-else is the variation of if statement. We attach a condition with if statement. If given condition is true then if block code will executed and if given condition is false then else block code will executed.

Syntax of if –else:-

```
if(Condition)
{
//if block code
}
else
{
//else block code
}
```

## Example Application:-

```
import java.util.Scanner;
class DecisionControlDemo2
{
public static void main(String [] args)
{
int a,b;
Scanner sc=new Scanner(System.in);
System.out.print(“Enter two numbers : ”);
a=sc.nextInt();
b=sc.nextInt();
if(a>b)
{
System.out.println(“Greatest No.=”+a);
}
else
{
System.out.println(“Greatest No.=”+b);
}
}
}
```

**O/P:-**

```
Enter two numbers : 100 200
Greatest No.=200
```

## Example Application:-

Develop a program in java to find roots of quadratic equation $ax^2 +bx+c=0$.

```
import java.util.Scanner;
import java.util.Math;
class Quad
{
public static void main(String [] args)
{
double a,b,c;
Scanner sc=new Scanner(System.in);
System.out.print(“Enter the value of a : ”);
a=sc.nextDouble();
System.out.print(“Enter the value of b : ”);
b=sc.nextDouble();
System.out.print(“Enter the value of c : ”);
c=sc.nextDouble();
double dis=Math.pow(b,2)-4*a*c;
If(dis<0)
{
System.out.println(“Roots are imaginary”);
}
else
{
double root1=(-b+Math.sqrt(dis))/(2*a);
double root1=(-b-Math.sqrt(dis))/(2*a);
System.out.println(“Root 1=”+root1);
System.out.println(“Root 2=”+root2);
}
}
}
```

**O/P 1:-**

```
Enter the value of a : 1
Enter the value of b : 2
Enter the value of c : 3
Roots are imaginary
```

**O/P 2:-**

```
Enter the value of a : 1
Enter the value of b : -2
Enter the value of c : 1
Root 1=1.0
Root 2=1.0
```

## Example Application:-

Develop a program in java to check given number is even or odd.

```
import java.util.Scanner;
class DecisionControlDemo3
{
public static void main(String [] args)
{
Scanner sc=new Scanner(System.in);
System.out.print("Enter a number : ");
int n=sc.nextInt();
if(n%2==0)
System.out.println("The No. "+n+" is even");
else
System.out.println("The No. "+n+" is odd");
}
}
```

**Nested if – else:-** If "if-else" construct is used inside if block or inside else block or inside both blocks then it is called nested if - else.

**Use of if-else construct inside if block:-**

```
if(condition1)
{
if(condition2)
{
//code 1
}
else
{
//code 2
}
```

```
}
else
{
//code 3
}
```

**Use of if-else construct inside else block:-**

```
if(condition1)
{
//code 1
}
else
{
if(condition2)
{
//code 2
}
else
{
//code 3
}
}
```

**Use of if-else construct inside both blocks:-**

```
if(condition1)
{
if(condition2)
{
//code 1
}
else
{
//code 2
}
}
else
{
if(condition3)
{
//code 3
```

```
}
else
{
//code 4
}
}
```

## Example Application:-

**Develop a program in java to find greatest number in three numbers using nested if – else.**

```
import java.util.Scanner;
class DecisionControlDemo3
{
public static void main(String [] args)
{
int x,y,z;
Scanner sc=new Scanner(System.in);
System.out.println(“Enter three numbers”);
x=sc.nextInt();
y=sc.nextInt();
z=sc.nextInt();
if(x>y)
{
if(x>z)
{
System.out.println(“Greatest No.=”+x);
}
else
{
System.out.println(“Greatest No.=”+z);
}
}
else
{
if(y>z)
{
System.out.println(“Greatest No.=”+y);
}
else
{
System.out.println(“Greatest No.=”+z);
}
```

```
}}}
```

## Ladder if – else:-

If you have multiple conditions and you want to execute the code based on those conditions then you can use ladder if – else .

**The syntax of if – else ladder is given below:-**

```
if(condition1)
{
//code 1
}
else if(condition2)
{
//code 2
}
else if(condition3)
{
//code 3
}
else
{
//code 4
}
```

## Example Application:-

**Develop a program in java to calculate electricity bill. Take number of units consumed by user and based on units calculate the bill. The parameters are given below:-**

| Unit range | Charge per unit |
|---|---|
| 1-150 | 2.40 Rs./Unit |
| For next 151-300 | 3.00 Rs./Unit |
| For next more than 300 | 3.20Rs./Unit |

```
import java.util.Scanner;
class ElectricityBill
{
public static void main(String [] args)
{
int unit;
double bill=0.0;
```

```
Scanner sc=new Scanner(System.in);
System.out.print("Enter the no. of units consumed : ");
unit=sc.nextInt();
if(unit<=150)
{
bill=unit*2.40;
}
else if(unit>150 && unit<=300)
{
bill=(150*2.40)+(unit-150)*3.00;
}
else
{
bill=(150*2.40)+(150*3.00)+(unit-300)*3.20;
}
System.out.println("Your bill="+bill);
}
}
```

**switch:-** switch is a keyword in java which works as case control. It is used to make menu based program.

1) Switch statement is used to declare multiple selections.

2) Inside the switch It is possible to declare any number of cases but is possible to declare only one default.

3) Switch is taking the argument the allowed arguments are **a. byte b. short c. int d.char e.String(allowed in 1.7 version)**

4) Float and double and long is not allowed for a switch argument because these are having more number of possibilities (float and double is having infinity number of possibilities) hence inside the switch statement it is not possible to provide float and double and long as a argument.

5) Based on the provided argument the matched case will be executed if the cases are not matched default will be executed.

**Syntax:-**

```
switch(argument)
{
case    label1 :
 sop(“ “);break;
 case   label2  :
sop(“ “);break;
|
|
 default :  sop(“  “); break;
 }
```

**Eg-1:Normal input and normal output.**

```
class Test
{
public static void main(String[] args)
{
int a=10;
switch (a)
{
case 10:System.out.println("Brijesh");  break;
case 20:System.out.println("Rohit");  break;
case 30:System.out.println("Satyam"); break;
 default:System.out.println("Rajeev");  break;
 }
 }
 }
```

**Ex-2:-Inside the switch the case labels must be unique; if we are declaring duplicate case labels the compiler will raise compilation error “duplicate case label”.**

```
class Test
{
public static void main(String[] args)
{
int a=10;
switch (a)
{
case 10:System.out.println("Brijesh");  break;
case 10:System.out.println("Satyam");  break;
```

```
case 30:System.out.println("Rohit"); break;
default:System.out.println("Rajeev");  break;
}
}
}
```

**Ex-3:Inside the switch for the case labels it is possible to provide expressions(10+10+20 , 10*4 , 10/2).**

```
class Test
{
public static void main(String[] args)
{
int a=100;
switch (a)
{
case 10+20+70:System.out.println("Brijesh");  break;
 case 10+5:System.out.println("Satyam");  break;
 case 30/6:System.out.println("Rohit");  break;
default:System.out.println("Yashi");   break;
}
}
}
```

**Eg-4:- Inside the switch the case label must be constant values. If we are declaring variables as a case labels the compiler will show compilation error "constant expression required".**

```
class Test
{
public static void main(String[] args)
{
int a=10; int b=20; int c=30;
switch (a)
{
case a:System.out.println("Brijesh");  break;
case b:System.out.println("Satyam");  break;
case c:System.out.println("Rohit");  break;
default:System.out.println("Yashi");  break;
}
}}
```

**Ex -5:-inside the switch independent statements are not allowed. If we are declaring the statements that statement must be inside the case or default.**

```
public class Test
{
public static void main(String[] args)
{
int x=10;
switch(x)
{
System.out.println("Hello World");
}
}
}
```

**Ex-6:- Internal conversion of char to integer.**

**Unicode values a-97 A-65**

```
class Test
{
public static void main(String[] args)
{
int a=65;
switch (a)
{
case 66: System.out.println("10"); break;
case 'A':System.out.println("20"); break;
 case 30:System.out.println("30"); break;
default: System.out.println("default"); break;
}
}
}
```

**Ex -7: internal conversion of integer to character.**

```
class Test
{
public static void main(String[] args)
{
char ch='d';
switch (ch)
{
case 100:System.out.println("10"); break;
case 'A':System.out.println("20"); break;
case 30:System.out.println("30"); break;
default: System.out.println("default"); break;
}
}
}
```

## Example Application:-

**Develop a program in java to convert temperature from $^0$c to $^0$f and $^0$f to $^0$c based on user choice (Temperature Convertor).**

```
import java.util.Scanner;
class TempConv
{
public static void main(String [] args)
{
double c,f;
int ch;
Scanner sc=new Scanner(System.in);
System.out.println(“Enter 1 for c to f”);
System.out.println(“Enter 2 for f to c”);
ch=sc.nextInt();
switch(ch)
{
case 1:
System.out.print(“Enter temperature in c : ”);
c=sc.nextDouble();
f=(9*c)/5+32;
System.out.println(“Temperature in f=”+f);
break;
case 2:
System.out.print(“Enter temperature in f : ”);
```

```
f=sc.nextDouble();
c=(f-32)*5/9;
System.out.println("Temperature in c="+c);
break;
default:
System.out.println("Invalid choice");
break;
}
}
}
```

**O/P 1:-**

```
Enter 1 for c to f
Enter 2 for f to c
1
Enter temperature in c : 12
Temperature in f=53.6
```

**O/P 2:-**

```
Enter 1 for c to f
Enter 2 for f to c
1
Enter temperature in c : 12
Temperature in f=53.6
```

**Loop Controls:-** If you have a block of code which you want to execute repeatedly then you can use a loop control. In java there are four types of loop controls in java:-

i.) while
ii.) for
iii.) do – while
iv.) for each

**while Loop:-** while is a keyword which works as a loop control. While is an entry control. The syntax of while loop is given below:-

```
Initialization of loop counter;
while(Condition)
{
//Body of Loop
Updation of loop counter;
}
```

## Example Application:-

```
class Test
{
public static void main(String[] args)
{
int i=0;
while (i<10)
{
System.out.println("Brijesh");
i++;
}
}
}
```

**Ex :- compilation error unreachable statement**

```
class Test
{
public static void main(String[] args)
{
int i=0;
while (false)
{
//unreachable statement
System.out.println("Brijesh");
i++;
}
}
}
```

**Example Application: -** Develop a program in java to generate series of even numbers from 1- 100.

```
class Test
{
public static void main(String [] args)
{
int i=1;
while(i<=100)
{
if(i%2==0)
{
System.out.print(i+"\t");
}
i++;
}
}
}
```

**Example Application:-** Develop a program in java to find sum of digits of given number.

```
import java.util.Scanner;
class Test
{
public static void main(String [] args)
{
int n;          //The variable which store the number
int sum=0;      //The variable which stores the result (Sum of digits)
int r;          //The variable which stores the result
Scanner sc=new Scanner(System.in);
System.out.print("Enter the number to find sum of digits : ");
n=sc.nextInt();
while(n>0)
{
r=n%10;
sum=sum+r;
n=n/10;
}
System.out.println("Sum of digits = "+sum);
}
}
```

**Example Application:-** Develop a program in java to find factorial of given number.

```
import java.util.Scanner;
class Test
{
public static void main(String [] args)
{
int n;
int f=1;
Scanner sc=new Scanner(System.in);
System.out.print("Enter the number to find factorial : ");
n=sc.nextInt();
while(n>0)
{
f=f*n;
n--;
}
System.out.println("Factorial = "+f);
}
}
```

**For Loop:-** for is a keyword which works as loop control. The for is also entry control. The working of for loop is same as while loop. But syntax is different.

**Syntax of for loop:-**

```
for (initialization ;condition  ;increment/decrement )
{
//Body of loop
}
```

**Initialization part:-** Initialization part it is possible to take the singe initialization it is not possible to take the more than one initialization.

**Ex1: Inside the for loop initialization part is optional.**

```
class Test
{
public static void main(String[] args)
{
int i=0;
for (;i<10;i++)
{
System.out.println("Softpro India");
}
}
}
```

**Ex 2:- Instead of initialization it is possible to take any number of System.out.println("Soft") statements and each and every statement is separated by camas(,) .**

```
class Test
{
public static void main(String[] args)
{
int i=0;
for (System.out.println("Soft");i<10;i++)
{
System.out.println("Brijesh");
}
}
}
```

**Example Application:-** Develop a program in java to generate Fibonacci Sequence upto n terms, where the value of n is entered by user.

```
import java.util.Scanner;
class Test
{
public static void main(String [] args)
{
int n1=0;                //First Term
int n2=1;                //Second Terms
int n3;                  //Third Term
int n;                   //No. of terms
int i;                   //Loop Counter
Scanner sc=new Scanner(System.in);
System.out.print("How many terms ? ");
n=sc.nextInt();
System.out.println("Fibonacci Sequence");
System.out.println(n1);
System.out.println(n2);
for(i=1;i<=n-2;i++)
{
n3=n1+n2;
System.out.println(n3);
n1=n2;
n2=n3;
}
}
}
```

**.Nested for – loop:-** If you use a for loop inside another for loop then it is called as nested for loop.

**Syntax of Nested for – loop:-**

```
for(initialization;condition;updation) {
//Code
for(initialization;condition;updation) {
//Code
}
//Code
}
```

**Example Application:-** Develop a program in java to generate series of prime numbers from 1-100.

```
import java.util.Scanner;
class  Prime
{
public static void main(String [] args)
{
int i,j,c=0;
System.out.println("Series of prime numbers from 1 to 100");
for(i=1;i<=100;i++)
{
c=0;
for(j=1;j<=i;j++)
{
if(i%j==0)
{
c++;
}
}
if(c==2)
System.out.print(i+"\t");
}
}
}
```

**do-while loop:-** do-while is a loop control, which works as exit control. In do-while loop the condition is tested at exit point i.e. after execution of code. We use do-while when we need to execute the code at least one time either condition is true or false.

## Syntax of do-while loop:-

```
Initialization of loop counter;
do
{
//Code
Updation of loop counter;
}
While (Condition);
```

**Example :-**

```
class Test
 public static void main(String[] args)
{
 int i=0;
do
 {
System.out.println("Softpro");
i++;
}
while (i<10);
}
}
```

**Example :- unreachable statement**

```
class Test
{
public static void main(String[] args)
{
int i=0;
do
{
 System.out.println("Brijesh");
 }
while (true);
System.out.println("Softpro India");//unreachable statement
}
}
```

**Example :-**

```
class Test
{
public static void main(String[] args)
{
int i=0;
do
{
System.out.println("Brijesh");
```

```
}
while (false);
System.out.println("Softpro India");
}
}
```

**Transfer statements:-**By using transfer statements we are able to transfer the flow of execution from one position to another position.

1. break 2. continue 3. return 4. try

**break:-** Break is used to stop the execution. We are able to use the break statement only two places.

**a. Inside the switch statement.**

**b. Inside the loops.**

if we are using any other place the compiler will generate compilation error message **" break outside switch or loop".**

**Example :-break means stop the execution come out of loop.**

```
class Test
{
public static void main(String[] args)
{
for (int i=0;i<10;i++)
{
if (i==5)
break;
System.out.println(i);
}
}
}
```

**Example :-**if we are using break outside switch or loops the compiler will raise compilation error **"break outside switch or loop"**

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

**continue:-(skip the current iteration and it is continue the rest of the iterations normally)**

```
class Test
{
public static void main(String[] args)
{
for (int i=0;i<10;i++)
{
if (i==5)
continue;
System.out.println(i);
}
}
}
```

## Technical Tasks

1. Write a java program to convert given number of days to a measure of time given in years, weeks and days. For example 375 days is equal to 1 year 1 week and 3 days (ignore leap year).
2. Write a Java program to accept a figure code and find the areas of different geometrical figures such as circle, square, rectangle etc. using switch.
3. Write a Java program to convert the given binary number into decimal.
4. Write a Java program to generate the series of prime numbers in range 1 – 100.
5. Write a Java program to check the given number is Armstrong or not.
6. Develop a program in java which take a number as input and print the table of that number in following manner:-

   2*1=2

   2*2=4

and so on.

7. Develop a program in java to display the size of fundamental data types.
8. Develop a program in java to find sum of digits of given number.

## Concept of array in Java:-

- Arrays are used to represent group of elements as a single entity but these elements are homogeneous & fixed size.
- The size of Array is fixed it means once we created Array it is not possible to increase and decrease the size.
- Array in java is index based first element of the array stored at 0 index.

## Advantages of array:-

- ✓ Instead of declaring individual variables we can declare group of elements by using array it reduces length of the code.
- ✓ We can store the group of objects easily & we are able to retrieve the data easily.
- ✓ We can access the random elements present in the any location based on index.
- ✓ Array is able to hold reference variables of other types.

| 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

## Different ways to declare an array:-

int[] values;

int []values;

int values[];

## Declaration& instantiation & initialization :-

**Approach 1:-** int a[]={10,20,30,40}; **//declaring, instantiation, initialization**

**Approach 2:-** int[] a=new int[100]; **//declaring, instantiation**

a[0]=10; **//initialization**

a[1]=20;

;;;;;;;;;;;;;

;;;;;;;;;;;;;

a[99]=40;

**// declares an array of integers**

```
int [] anArray;
```

**// allocates memory for 10 integers**

```
anArray = new int[10];
```

**// initialize first element**

```
anArray[0] = 10;
```

**// initialize second element**

```
anArray[1] = 20;
```

**// and so forth**

```
anArray[2] = 30;    anArray[3] = 40;    anArray[4] = 50;    anArray[5] = 60;
anArray[6] = 70;    anArray[7] = 80;    anArray[8] = 90;    anArray[9] = 100;
```

**Example :- taking array elements from dynamic input by using scanner class.**

```
import java.util.*;
class Test  {
public static void main(String[] args)   {
 int[] a=new int[5];
 Scanner s=new Scanner(System.in);
System.out.println("enter values");
for (int i=0;i<a.length;i++)   {
System.out.println("enter "+i+" value");
 a[i]=s.nextInt();
 }
for (int a1:a){
System.out.println(a1);
}
}
}
```

**Example :- find the sum of the array elements.**

```
class Test  {
public static void main(String[] args)   {
int[] a={10,20,30,40};
int sum=0;
for (int a1:a)   {
sum=sum+a1;
}
System.out.println("Array Element sum is="+sum);
}
}
```

**Method parameter is array & method return type is array:-**

```
class Test  {
static void m1(int[] a) //method parameter is array
{
for (int a1:a)
{
System.out.println(a1);
}
}
static int[] m2() //method return type is array
{
System.out.println("m1 method");
return new int[]{100,200,300};
}
public static void main(String[] args)   {
Test.m1(new int[]{10,20,30,40});
int[] x  = Test.m2();
for (int x1:x)
{
System.out.println(x1);
}
}
}
```

**Example :- copy the data from one array to another array**

```
class Test  {
public static void main(String[] args)   {
int[] copyfrom={10,20,30,40,50,60,70,80};
int[] copyto = new int[7];
System.arraycopy(copyfrom,1,copyto,0,7);
 for (int cc:copyto)
{
System.out.println(cc);
 }
}
}
```

**Example:-process of adding different types  Objects in Object array**

**Test.java:-**

```
class Test  {
public static void main(String[] args)   {
Object[] a= new Object[6];
a[0]=new Emp(111,"Brijesh");
a[1]=new Integer(10);
a[2]=new Student(1,"Yashi");
 for (Object a1:a)
{
if (a1 instanceof Emp)
{
Emp e1 = (Emp)a1;
System.out.println(e1.eid+"---"+e1.ename);
}
if (a1 instanceof  Student)
{
Student s1 = (Student)a1;
System.out.println(s1.sid+"---"+s1.sname);
}
if (a1 instanceof Integer)
{
System.out.println(a1);
}
if (a1==null)
{
System.out.println(a1);
```

```
}
}
}
}
```

**Emp.java:**

```
class Emp
{
int eid;
String ename;
Emp(inteid,Stringename)
{
//conversion of local to instance
this.eid=eid;
this.ename=ename;
}
}
```

**Student.java:-**

```
class Student
{
int sid;
String sname;
Student(int sid,String sname)
{
//conversion of local to instance
this.sid=sid;
this.sname=sname;
}
}
```

## Example :-febonacci series

```
import java.util.Scanner;
class Test
{
public static void main(String[] args)
{
System.out.println("enter start series of febonacci");
 int x = new Scanner(System.in).nextInt();
int[] feb = new int[x];
feb[0]=0;
feb[1]=1;
 for (int i=2;i<x;i++)
{
 feb[i]=feb[i-1]+feb[i-2];
}
//print the data
for (int feb1 : feb)
{
System.out.print(" "+feb1);
}
}
}
```

## Pre-increment & post increment :-

**Pre-increment :- it increases the value by 1 then it will execute statement.**

**Post-increment :-it executes the statement then it will increase value by 1.**

```
class Test
{
public static void main(String[] args)   {
//post increment
 int a=10;
System.out.println(a);  //10
System.out.println(a++); //10
System.out.println(a);  //11
//pre increment
int b=20;
System.out.println(b);  //20
System.out.println(++b); //21
System.out.println(b);  //21
```

```
System.out.println(a++ + ++a + a++ + ++a);       //11   13   13   15
}
}
```

## Pre-decrement &post decrement :-

**Pre-decrement :- it decreases the value by 1 then it will execute statement.**

**Post-decrement :-it executes the statement then it will increase value by 1.**

```
class Test
{
public static void main(String[] args)
{
//post decrement
int a=10;
System.out.println(a);  //10
System.out.println(a--); //10
System.out.println(a);  //9
//post decrement
int b=20;
System.out.println(b);  //20
System.out.println(--b); //19
System.out.println(b);  //19
System.out.println(a-- + --a + a-- + --a);       //9   7  7   5
}
}
```

## Multi-dimensional Array:-

The two dimensional array have two subscripts one for rows and another one for column. The two dimensional array is sometimes called matrix.

**Declaration of two dimensional array:-**

int [][] x=new int[3][3];

**Example:- Take a matrix of 3*3 as input and display the matrix elements.**

```
import java.util.Scanner;
class Test{
public static void main(String [] args){
int x[][]=new int[3][3];
int i,j;
Scanner sc=new Scanner(System.in);
System.out.println("Enter a matrix of 3*3");
for(i=0;i<3;i++){
for(j=0;j<3;j++){
x[i][j]=sc.nextInt();
}
}
System.out.println("You have entered following matrix");
for(i=0;i<3;i++){
for(j=0;j<3;j++){
System.out.print(x[i][j]+" ");
}
System.out.print("\n");
}
}
}
```

## String Manipulation

**1) Java.lang.String**

**2) Java.lang.StringBuffer**

**3) Java.lang.StringBuilder**

**4) Java.util.StringTokenizer**

**Java.lang.String:-**

String is used to represent group of characters or character array enclosed with in the double quotes.

```
class Test  {
public static void main(String[] args)   {
String str="softpro";
System.out.println(str);
String str1=new String("softpro");
System.out.println(str1);
char[] ch={'s','o','f','t','p',’r’,’o’};
String str3=new String(ch);
System.out.println(str3);
}
}
```

**Case 1:-String vs StringBuffer**

String  &StringBuffer both classes are final classes present in java.lang package.

**Case 2:-String vsStringBuffer**

We are able to create String object in two ways.

1) Without using new operator  String str=”softpro”;

2) By using new operator  String str = new String(“softpro”);

We are able to create StringBuffer object only one approach by using new operator.
StringBuffer sb = new StringBuffer(“softpro”);

## Example:-

```
class Test  {
public static void main(String[] args)   {
 //two approaches to create a String object
String str1 = "softpro";
System.out.println(str1);
String str2 = new String("India");
System.out.println(str2);
  //one approach to create StringBuffer Object (by using new operator)
StringBuffersb = new StringBuffer("Softpro India");
System.out.println(sb);
}
}
```

## ==operator :-

- ✓ It is comparing reference type and it returns Boolean value as a return value.
- ✓ If two reference variables are pointing to same object then it returns true otherwise false.

## Example:-

```
class Test  {
public static void main(String[] args)   {
Test t1 = new Test();
Test t2 = new Test();
Test t3 = t1;
System.out.println(t1==t2); //false
System.out.println(t1==t3); //true
String str1="softpro";
String str2="softpro";
System.out.println(str1==str2); //true
String s1 = new String("softpro");
String s2 = new String("softpro");
System.out.println(s1==s2); //false
StringBuffer sb1 = new StringBuffer("softpro");
StringBuffer sb2 = new StringBuffer("softpro");
System.out.println(sb1==sb2); //false
}
}
```

## Case 3:- String

**Java.lang.String vs java.lang.StringBuffer:-**

- ✓ String is immutability class it means once we are creating String objects it is not possible to perform modifications on existing object. (String object is fixed object)
- ✓ StringBuffer is a mutability class it means once we are creating StringBuffer objects on that existing object it is possible to perform modification.

## Example :-

```
class Test  {
public static void main(String[] args)   {
//immutability class (modifications on existing content not allowed)
String str="Softpro";
str.concat("India");
System.out.println(str);
//mutability class (modifications on existing content possible)
StringBuffer sb = new StringBuffer("Softpro");
sb.append("India");
System.out.println(sb);
}
}
```

**concat( ) :-**

concat() method is combining two String objects and it is returning new String object.

public java.lang.String concat(java.lang.String);

## Example :-

```
class Test  {
public static void main(String[] args)   {
String str="softpro";
String str1 = str.concat("india");//concat() method return String object.
System.out.println(str);
System.out.println(str1);
}
}
```

## Example Application:- Develop a program in java to compare two strings for equality.

```
import java.util.Scanner;
class Test {
public static void main(String [] args) {
Scanner sc=new Scanner(System.in);
System.out.print(“Enter first string : ”);
String str1=sc.nextLine();
```

```
System.out.print(“Enter second string : ”);
String str2=sc.nextLine();
if(str1.equals(str2)==true)
System.out.println(“Both strings are equal”);
else
System.out.println(“Both strings are not equal”);
}
}
```

**Example Application:- Develop a program in java to search a pattern in a string.**

```
import java.util.Scanner;
class Test {
public static void main(String [] args) {
Scanner sc=new Scanner(System.in);
System.out.print(“Enter main string : ”);
String str=sc.nextLine();
System.out.print(“Enter subsstring : ”);
String substr=sc.nextLine();
if(str.contains(substr)==true)
System.out.println(“The substring : ”+substr+” is available in main string : ”+str);
else
System.out.println(“The substring is not available in main string”);
}
}
```

**Example Application:- Develop a program in java to check given string is palindrome or not.**

```
import java.util.Scanner;
class Test {
public static void main(String [] args) {
Scanner sc=new Scanner(System.in);
System.out.print(“Enter a string : ”);
String str=sc.nextLine();
String revstr=””;
for(int i=0;i<str.length();i++)
{
revstr=revstr+str.charAt(i)+””;
}
```

```
if(str.equals(revstr)==true)
System.out.println("String is palindrome");
else
System.out.println("String is non-palindrome");
}
}
```

**Example Application:- Develop a program in java to take a sentence as input. Find a word in sentence. Replace the word with another word in sentence.**

```
import java.util.Scanner;
class Test {
public static void main(String [] args){
String sentence, fw, rw;
Scanner sc=new Scanner(System.in);
System.out.print("Enter a sentence : ");
sentence=sc.nextLine();
System.out.print("Find what : ");
fw=sc.nextLine();
System.out.print("Replace with : ");
rw=sc.nextLine();
System.out.println("Modified sentence : "+sentence.replace(fw,rw));
}
}
```

**Example Application:- Develop a program in java to take the user name as input and display the short name.**

**E.g.I/P: Ajay Kumar Singh**

**O/P: A.K.Singh**

```
import java.util.Scanner;
class Test{
public static void main(String [] args){
Scanner sc=new Scanner(System.in);
System.out.print("Enter your name : ");
String name=sc.nextLine();
String shortname[]=name.spilt(" ");
System.out.print("Your short name : ");
```

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

## Technical Tasks

1. Develop a program in java which takes input from command line arguments. And display the elements.
2. Develop a program in java which takes five names as input and display those names in alphabetical order.
3. Develop a program in java to take ten numbers as input and store those numbers in array. Take a number and search it in list using binary search technique.
4. Develop a program in java to take ten number list display the largest and smallest number of the list.
5. Develop a program in java for matrix multiplication.
6. Develop a program in java which checks the given string is palindrome or not.
7. Develop a program to take user name as input and display the name in uppercase and lowercase letters.
8. Develop a program in java to take a decimal number as input and display it in binary, octal and hexa-decimal equivalent.

## Method in Java:-

- ✓ Methods are used to write the business logics of the project.
- ✓ Coding conversion of method is method name starts with lower case letter if method contains more than one word then every inner word starts with uppercase letter. Example:- post() , charAt() , toUpperCase() , compareToIgnoreCase()……etc
- ✓ There are two types of methods **1. Instance method 2. Static method**
- ✓ Inside the class it is possible to declare any number of instance methods &static methods based on the developer requirement.
- ✓ It will improve the reusability of the code and we can optimize the code.

Note :- Whether it is an instance method or static method the methods are used to provide business logics of the project.

**Instance method :-**

```
void m1()  //instance method
{
//body    instance area
}
```

Note: - for the instance members memory is allocated during object creation hence access the instance members by using object (reference-variable).

## Syntax:-

void m1() { } **//instance method**

Test t = new Test();

**Objectname.instancemethod( ); //calling  instance method**

t.m1( );

## static method:-

```
static void m1() //instance method
{
//body static method
}
```

**Note: -** for the static member's memory allocated during .class file loading hence access the static members by using class-name.

## Syntax:-

static void m2() { } **//static method**

**Classname.staticmethod( ); //  call static method by using class name**

Test.m2( );

**Syntax:- [modifiers-list] return-Type Method-name (parameters list) throws Exception**

Modifiers-list ---→ represent access permissions.---→ [optional]

Return-type ---→ functionality return value---→ [mandatory]

Method name ---→ functionality name ----→ [mandatory]

Parameter-list -----→ input to functionality ----→ [optional]

Throws Exception ---→ representing exception handling------→ [optional]

## Example:-

public void m1()

private int m2(int a,int b)

## Method Signature:-

Method Signature is nothing but name of the method and parameters list. Return type and modifiers list not part of a method signature.

## Syntax:-

Method-name(parameter-list)

## Ex:-

m1(int a)

m1(int a,int b)

Every method contains two parts.

1. Method declaration

2. Method implementation (logic)

## Ex:-

```
void m1()  ------>  method declaration
{
Body  (Business logic);  -----> method implementation
}
```

**Example-1 :- instance methods without arguments.**

Instance methods are bounded with objects hence call the instance methods by using object name(reference variable).

```
class Test  {
//instance methods
void spi() { System.out.println("Softpro India");   }
void slc() { System.out.println("Softpro Learning Center"); }
public static void main(String[] args)   {
Test t=new Test();
t.spi(); //calling of instance method by using object name [ t ]
t.slc();  //calling of instance method by using object name [ t ]
}
}
```

**Example-2:-instance methods with parameters.**

- ✓ If the method is taking parameters at that situation while calling that method must provide parameter values then only that method will be executed.
- ✓ Parameters of methods is nothing but inputs to method.
- ✓ While passing parameters number of arguments and argument order is important.

| | | |
|---|---|---|
| void m1(int a) | -->t.m1(10); | -->valid |
| void m2(int a,int b) | -->t.m2(10,'a'); | -->invalid |
| void m3(int a,char ch,float f) | -->t.m3(10,'a',10.6); | -->invalid |
| void m4(int a,char ch,float f) | -->t.m4(10,10,10.6); | -->invalid |
| void m5(int a,char ch,float f) | -->t.m3(10,'c'); | -->invalid |

```
class Test  {
//instance methods
void m1(int i,char ch)  //local variables
{
System.out.println(i+"-------"+ch);
}
void m2(double d ,String str) //local variables
{
System.out.println(d+"------"+str);
}
public static void main(String[] args)
{
Test t=new Test();
t.m1(10,'a');    //m1() method calling
t.m2(10.2,"softpro");   //m2() method calling
}
}
```

**Example-3 :- static methods without parameters.**

Static methods are bounded with class hence call the static members by using class name.

```
class Test  {
//static methods
static void m1()  {
System.out.println("m1 static method");
}
static void m2()  {
System.out.println("m2 static method");
}
public static void main(String[] args)  {
Test.m1();  //call the static method by using class name
Test.m2();  //call the static method by using class name
}
}
```

**Example -4 :-static methods with parameters.**

```
class Test {
//static methods
static void m1(String str,char ch,int a) //local variables
{
System.out.println(str+"---"+ch+"---"+a);
}
static void m2(boolean b1,double d)   //local variables
{
System.out.println(b1+"---"+d);
}
public static void main(String[] args)  {
Test.m1("softpro",'a',10);    //static m1() calling by using class name
Test.m2(true,10.5);  //static m2() calling by using class name
}
};
```

**Example 6:-For java methods it is possible to provide Objects as a parameters(in real time project level).**

**Case 1:- project code at student level.**

```
class X{}
class Emp{}
class Y{}
class Test  {
void m1(X x ,Emp e)  {
System.out.println("m1 method");
}
static void m2(int a,Y y)  {
System.out.println("m2 method");
}
public static void main(String[] args)   {
Test t = new Test();
X x = new X();
Emp e = new Emp();
t.m1(x,e);     //calling of instance method by using object
Y y = new Y();
Test.m2(10,y);  //calling of static method by using class-name
}
}
```

**Case 2: project code at realtime project level**

```
class X{}
class Emp{}
class Y{}
class Test  {
void m1(X x ,Emp e)//taking objects as a parameter
{
System.out.println("m1 method");
}
static void m2(int a,Y y) //taking objects as a parameter
{
System.out.println("m2 method");
}
public static void main(String[] args)   {
new Test().m1(new X(),new Emp());
Test.m2(10,new Y());
}
}
```

**Example-7 :- method vs. data- types**

- ✓ By default the numeric values are integer valuesbut to represent other format like byte, short perform typecasting.
- ✓ By default the decimal values are double values but to represent float value perform typecasting or use “f” constant.
- ✓ double d=10.5;  float f=20.5f;

```
class Test  {
void m1(byte a)  { System.out.println("Byte value-->"+a);  }
void m2(short b ) { System.out.println("short value-->"+b);  }
void m3(int c)  { System.out.println("int value-->"+c);  }
void m4(long d)  { System.out.println("long value is-->"+d); }
void m5(float e)  { System.out.println("float value is-->"+e); }
void m6(double f) { System.out.println("double value is-->"+f); }
void m7(char g)  { System.out.println("character value is-->"+g); }
void m8(boolean h) { System.out.println("Boolean value is-->"+h); }
public static void main(String[] args)   {
Test t=new Test();
```

```
t.m1((byte)10);  //by default 10 is int value
t.m2((short)20); //by default 20 is int value
t.m3(30);   t.m4(40);   t.m5(10.6f);  //by default 10.6  value is  double
t.m6(20.5);   t.m7('a');  t.m8(true);
}
}
```
✓

## Technical Tasks

1. Write a program in java to make two user derived functions **area()** and **peri()** which compute the area and perimeter of circle.
2. Develop a program in java to find factorial of given number using 'Recursion'
3. Develop a program in java to generate Fibonacci Sequence up to n terms using 'Recursion'.
4. Develop a program in java to make simple calculator using user defined functions.
5. Develop a program in java to make Currency Convertor using user-defined functions.

# OOPS (Object Oriented Programming System)

**Class:-** The Class is a container of variables, methods and constructors. Or The Class contains data members and member functions. The class is declared by using **"class"** keyword followed by class name. The body of class is enclosed within braces and terminated by semicolon (Optional in Java).

```
class class_name
{
//Body of class
};
```

**Example 1:- Use of public access specifier.**

```
class car
{
public String make;     //Data member
public String color;    //Data member
public int price;       //Data member
}
class Test
{
public static void main(String [] args)
{
Car c;   //Reference variable of class Car
c=new Car();   //Creation of object of class Car
c.make="Tata";
c.color="Silver";
c.price=650000;
System.out.println("Car Make-> "+c.make);
System.out.println("Car Color-> "+c.color);
System.out.println("Car Price-> "+c.price);
}
}
```

**Example 2:- Use of private access specifier.**

```
class Car
{
private String make;   //private data member
private String color;   //private data member
private int price;        //private data member
public void setCar (String make, String color, int price)
{
this.make=make;
this.color=color;
this.price=price;
}
public void display()
{
System.out.println("Car make-> "+this.make);
System.out.println("Car color-> "+this.color);
System.out.println("Car price-> "+this.price);
}
};
class Test
{
public static void main(String [] args)
{
Car c=new Car();
c.setCar("Tata","Silver",650000);
c.display();
}
}
```

**Note:- The private data members are not directly accessible outside of class. These are accessible via public member functions.**

## Constructor:-

The Constructor is a special member function which is used to initialize final data members.

**Rules to declare constructor:-**

1) Constructor name class name must be same.

2) Constructor is able to take parameters.

3) Constructor not allowed explicit return type (return type declaration not possible).

**There are two types of constructors:-**

1) Default Constructor (provided by compiler).

2) User defined Constructor (provided by user) or parameterized constructor.

**Default Constructor:-**

1) If we are not write constructor for a class then compiler generates one constructor for you that constructor is called default constructor. And it is not visible in code.

2) Compiler generates Default constructor inside the class when we are not providing any type of constructor (0-arg or parameterized).

3) The compiler generated default constructor is always 0-argumnetconstructor with empty implementation (empty body).

**Application before compilation :-**

```
class Test  {
void m1() {
System.out.println("m1 method");
}
public static void main(String[] args)
{
//at object creation time 0-arg constructor executed
Test t = new Test();   t.m1();
}
}
```

In above application when we create object by using new keyword **"Test t = new Test()"** then compiler is searching **"Test()"**constructor inside the since not there hence compiler generate default constructor at the time of compilation.

**Application after compilation :-**

```
class Test  {
void m1() {   System.out.println("m1 method"); }
//default constructor generated by compiler
Test()    {
}
public static void main(String[] args)
{
//object creation time 0-arg constructor executed
Test t = new Test();
 t.m1();
 }
}
```

In above example at run time JVM execute compiler provide default constructor during object creation.

## Example-3:-

- Inside the class if we declaring at least one constructor (either 0-arg or parameterized) the compiler won't generate default constructor.
- if we are trying to compile below application the compiler will generate error message "cannot find symbol " .
- Inside the class if we are declaring at least one constructor the compiler won't generate default constructor.

```
class Test  {
Test(int i)  { System.out.println(i);  }
Test(int i,String str)  { System.out.println(i);
System.out.println(str);  }
public static void main(String[] args)
{
Test t1=new Test(); // in this line compiler is searching 0-arg cons but not available
Test t2=new Test(10);
Test t3=new Test(100,"Softpro");
}
}
```

**Example:- default constructor execution vs. user defined constructor execution.**

**Case 1:- default constructor execution process.**

```
class Employee
{
//instance variables
int eid;
String ename;
double esal;
void display()
{
//printing instance variables values
System.out.println("****Employee details****");
System.out.println("Employee name :-->"+ename);
System.out.println("Employee eid :-->"+eid);
System.out.println("Employee sal :-->"+esal);
}
public static void main(String[] args)
{
// during object creation 0-arg cons executed then values are assigned
Employee e1 = new Employee();
e1.display();
}
}
```

**D:\morn11>javac Employee.java**

**D:\morn11>java Employee**

******Employee details******

**Employee name :-->null**

**Employee eid :-->0**

**Employee sal :-->0.0**

**Note: - in above example during object creation time default constructor is executed with empty implementation and initial values of instance variables (default values) are printed.**

**Case 2:- user defined o-argument constructor execution process.**

```
class Employee
{
//instance variables
int eid;
String ename;
double esal;
Employee()//user defined 0-argument constructor
{
//assigning values to instance values during object creation
eid=111;
ename="Brijesh";
esal =60000;
}


void display()
{
//printing instance variables values
System.out.println("****Employee details****");
System.out.println("Employee name :-->"+ename);
System.out.println("Employee eid :-->"+eid);
System.out.println("Employee esal :-->"+esal);
}
public static void main(String[] args)
{
// during object creation 0-arg cons executed then values are assigned
 Employee e1 = new Employee();
e1.display();//calling display method
}
}
```

**Compilation & execution process:-**

**D:\morn11>javac Employee.java**

**D:\morn11>java Employee**

******Employee details******

**Employee name :-->Brijesh**

**Employee eid :-->111**

**Employee esal :-->60000.0**

**Note: - in above example during object creation user provided 0-arg constructor executed used to initialize some values to instance variables.**

## The problem with above approach:-

When we create two employee objects but every object same values are initialized.

```
Emp e1 = new Emp();

e1.display();

Emp e2 = new Emp();

e2.display();
```

```
D:\morn11>java Employee

****Employee details****

Employee name :-->Brijesh

Employee eid :-->111

Employee esal :-->60000.0

Employee name :-->Brijesh

Employee eid :-->111

Employee esal :-->60000.0
```

**To overcome above limitation just use parameterized constructor to initialize different values.**

## Case 3:- user defined parameterized constructor execution. User defined parameterized constructors:-

- Inside the class if the default constructor is executed means the initial values of variables only printed.
- To overcome above limitation inside the class we are declaring user defined 0-argument constructor to assign some values to instance variables but that constructor is able to initialize the values only for single object.
- To overcome above limitation declare the parameterized constructor and pass the different values during different objects creation.
- Parameterized constructor is nothing but the constructor is able to parameters.

## Example :-

```
class Employee
{
//instance variables
int eid;
String ename;
double esal;
Employee(int eid,String ename,double esal) //local variables
{
//conversion (passing local values to instance values)
this.eid = eid;
this.ename = ename;
this.esal = esal;
}
void display()
{
//printing instance variables values
System.out.println("****Employee details****");
System.out.println("Employee name :-->"+ename);
System.out.println("Employee eid :-->"+eid);
System.out.println("Employee esal :-->"+esal);
}
public static void main(String[] args)
{
// during object creation parameterized  constructor executed
Employee e1 = new Employee(111,"Brijesh",60000);
e1.display();
Employee e2 = new Employee(222,"Rohit",70000);
e2.display();
Employee e3 = new Employee(333,"Yashi",80000);
e3.display();
}
}
```

**Compilation & execution process:-**

**D:\morn11>javac Employee.java**

**D:\morn11>java Employee**

******Employee details******

**Employee name :-->Brijesh**

**Employee eid :-->111**

**Employee esal :-->60000.0**

******Employee details******

**Employee name :-->Rohit**

**Employee eid :-->222**

**Employee esal :-->70000.0**

******Employee details******

**Employee name :-->Yashi**

**Employee eid :-->333**

**Employee esal :-->80000.0**

## Technical Tasks

1. Design a class to represent a bank account. Include the following members:-
   Data Members
   i.)Name of the depositor
   ii.)Account Number
   iii.)Type of Account
   iv.)Balance Amount in Account
   Methods
   i.) To assign initial values
   ii.) To deposit an amount

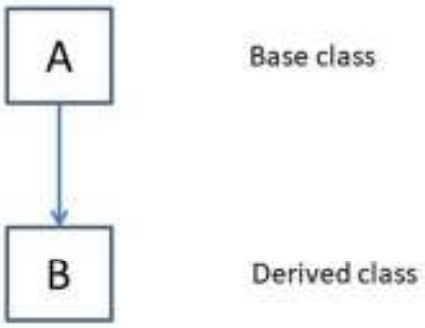
 

iii.) To Withdraw an amount after checking balance
iv.) To display the name and balance.

2. Develop a class named Rectangle with private data members length and width. Make a parameterized Constructor to initialize data members. Now make two methods rectarea() and rectperi() to calculate area and perimeter. Test the class Rectangle.
3. Develop a program in Java to ask user enter his name or any English word. Display total value of his name, total value means sum of all alphabets assuming A=1,B=2,……..Z=26. Test it with word "attitude".
4. Develop a class named Interest with private data members p,n,r of float type. Make a parameterized Constructor to initialize data members. Now make a method simpleInterest() to calculate simple interest. Now test the class Interest.

## Inheritance

Inheritance is a feature of Object Oriented Programming. In Inheritance you can create a new class by using existing class. The existing class is called base class and new created class is called derived class. The concept of Inheritance is also called 'Reusability'.




### Syntax in Java:-

```
class A            //Base Class
{
……..;
……..;
}
class B extends A        //Derived Class
{
……..;
……..;
}
```

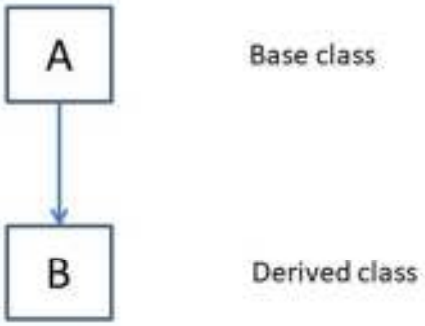
 

## Types of Inheritance in Java:-

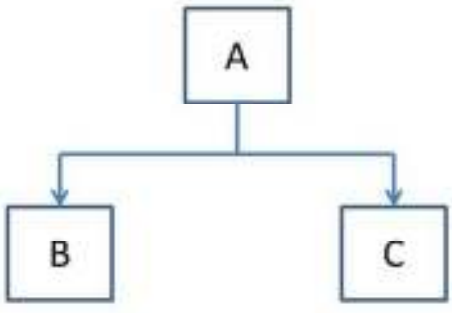
In Java there are three types of Inheritance supported in Java:-

i.) Single/ Simple Inheritance
ii.) Hierarchical Inheritance
iii.) Multi – level Inheritance

**Single Inheritance:-** In Single Inheritance there is a single base class and single derived class.



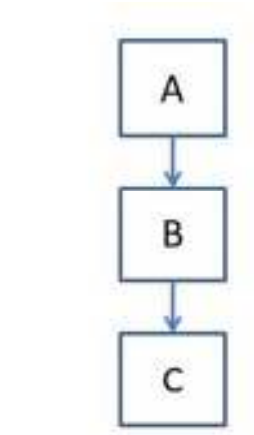

**Hierarchical Inheritance:-** In Hierarchical Inheritance there is a single base class and multiple derived class.




Hierarchical Inheritance

## Multi-level Inheritance:-




Multi-level Inheritance

## Example Application:- Demonstration of Single Inheritance.

```
class Rundog
{
public void bark()
{
System.out.println("Tommy……..");
System.out.println("Bho…….Bho……..");
}
}
class Bulldog extends Rundog
{
public void grawl()
{
System.out.println("Tuffy……..");
System.out.println("Gurr……Gurr……");
}
}
class Test
{
public static void main()
{
Bulldog dog=new Bulldog();
dog.bark();
```

```
dog.grawl();
}}
```

**Example Application: -** A program in java to demonstrate Hierarchical Inheritance.

```
import java.util.Scanner;
class Shape
{
protected int s;          //protected data member
public void setValue(int x) //public method to initialize data member
{
s=x;
}
}
class Cube extends Shape
{
public int volume()
{
return (s*s*s);
}
}
class Square extends Shape
{
public int area()
{
return(s*s);
}
}
class Test
{
public static void main(String [] args)
{
Scanner sc=new Scanner(System.in);
int x;
System.out.print(“Enter side of cube : ”);
x=sc.nextInt();
Cube cu=new Cube();
cu.setValue(x);
System.out.println(“Volume of cube : ”+cu.volume());
System.out.print(“Enter side of square : ”);
x=sc.nextInt();
Square sq=new Square();
sq.setValue(x);
System.out.println(“Area of square : ”+sq.area());
}
```

```
}
```

**O/P:-**
**Enter side of cube : 10**
**Volume of cube : 1000**
**Enter side of square : 10**
**Area of square : 100**

**Example Application:- A program in java to demonstrate Multi-level inheritance.**

```
class A
{
public void showA()
{
System.out.println("This message from class A");
}
}
class B extends A
{
public void showB()
{
System.out.println("This message from class B");
}
}
class C extends B
{
public void showC()
{
System.out.println("This message from class C");
}
public static void main(String [] args)
{
C c=new C();
c.showA();
c.showB();
c.showC();
}
}
```

## Polymorphism:-

The term “Polymorphism” means “One Thing Many Forms”. There are two types of Polymorphism in Java:-

i.) Compile Time Polymorphism (Overloading)
ii.) Run Time Polymorphism (Overriding)

**Method Overloading:-** In Java you can give the same name to multiple methods of same class, but their parameters must be different.

## Compile time polymorphism [Overloading]:-

1) If java class allows two methods with same name but different number of arguments such type of methods are called overloaded methods.

2) We can overload the methods in two ways in java language

a. By passing different number of arguments to the same methods.

**void m1(int a){}**

**void m1(int a,int b){}**

b. Provide the same number of arguments with different data types.

**void m1(int a){}**

**void m1(char ch){}**

3) If we want achieve overloading concept one class is enough.

4) It is possible to overload any number of methods in single java class.

## Types of overloading:-

a. Method overloading (explicitly by the programmer)

b. Constructor overloading (explicitly by the programmer)

c. Operator overloading implicitly by the JVM(‘+’ addition& concatenation)

## Method overloading Example:-

```
class Test  {
//below three methods are overloaded methods.
void m1(char ch) {System.out.println(" char-arg constructor "); }
void m1(int i)  {System.out.println("int-arg constructor "); }
void m1(int i,int j) {System.out.println(i+j);   }
public static void main(String[] args)
{
Test t=new Test();  //three methods execution decided at compilation time
t.m1('a');
t.m1(10);
t.m1(10,20);
}
}
```

## Example :- overloaded methods vs. all data types.

```
class Test  {
void m1(byte a)   {     System.out.println("Byte value-->"+a);     }
void m1(short a)  {    System.out.println("short value-->"+a);    }
void m1(int a)     {    System.out.println("int value-->"+a);        }
void m1(long a)    {     System.out.println("long value is-->"+a); }
void m1(float f)    {    System.out.println("float value is-->"+f); }
void m1(double d) {    System.out.println("double value is-->"+d);  }
void m1(char ch)    {    System.out.println("character value is-->"+ch);  }
void m1(boolean b) {  System.out.println("boolean value is-->"+b);   }
void sum(int a,int b)  { System.out.println("int arguments method");   System.out.println(a+b);  }
void sum(long a,long b)  { System.out.println("long arguments method");
System.out.println(a+b);  }
public static void main(String[] args)   {
Test t=new Test();   t.m1((byte)10);   t.m1((short)20);  t.m1(30);  t.m1(40);   t.m1(10.6f);
t.m1(20.5);  t.m1('a');  t.m1(true);   t.sum(10,20);   t.sum(100L,200L);
}
}
```

**Constructor Overloading:-** The class contains more than one constructors with same name but different arguments is called constructor overloading.

```
class Test  {
//overloaded constructors
Test()  { System.out.println("0-arg constructor");  }
Test(int i) { System.out.println("int argument constructor"); }
Test(char ch,int i){ System.out.println(ch+"-----"+i);   }
public static void main(String[] args)   {
Test t1=new Test(); //zero argument constructor executed.
Test t2=new Test(10); // one argument constructor executed.
Test t3=new Test('a',100);//two argument constructor executed.
}
}
```

## Operator overloading:-

- ✓ One operator with different behavior is called Operator overloading .
- ✓ Java is not supporting operator overloading but only one overloaded in java language is '+'.
    - If both operands are integer + perform addition.
    - If at least one operand is String then + perform concatenation.

## Example:-

```
class Test  {
public static void main(String[] args)
{
int a=10;
int b=20;
System.out.println(a+b); //30 [addition]
System.out.println(a+"Brijesh"); //10Brijesh [concatenation]
}
}
```

## Runtime polymorphism [Method Overriding]:-

1) If we want to achieve method overriding we need two class with parent and child relationship.

2) The parent class method contains some implementation (logics).

a. If child is satisfied use parent class method.

b. If the child class not satisfied (required own implementation) then override the method in Child class.

3) A subclass has the same method as declared in the super class it is known as method overriding.

**The parent class method is called ===> overridden method**

**The child class method is called ===> overriding method**

**While overriding methods must fallow these rules:-**

1) While overriding child class method signature & parent class method signatures must be same otherwise we are getting compilation error.

2) The return types of overridden method & overriding method must be same.

3) While overriding the methods it is possible to maintains same level permission or increasing order but not decreasing order, if you are trying to reduce the permission compiler generates error message “attempting to assign weaker access privileges ”.

4) You are unable to override final methods. (Final methods are preventing overriding).

5) While overriding check the covariant-return types.

6) Static methods are bounded with class hence we are unable to override static methods.

7) It is not possible to override private methods because these methods are specific to class.

**Example-1 :-method Overriding**

```
class Parent //parent class
{
void property()  {System.out.println("money+land+hhouse");}
void marry()  {System.out.println("black girl"); }//overridden method
};
class Child extends Parent//child class
{
void marry() {System.out.println("white girl/red girl");} //overriding method
public static void main(String[] args)
{
Child c=new Child();
c.property();
c.marry();
Parent p=new Parent();
p.property();
p.marry();
} };
```

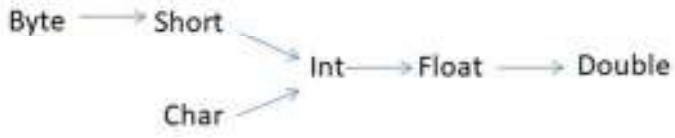
**Type-casting:-** The process of converting data one type to another type is called type casting. There are two types of type casting

1. Implicit typecasting /widening/up casting

2. Explicit type-casting (narrowing)/down casting

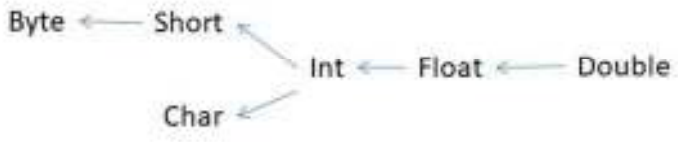
## Type casting chart:-

## Upcasting:-



## Downcating:-




When we assign higher value to lower data type range then compiler will rise compiler error "possible loss of precision" but whenever we are type casting higher data type-lower data type compiler won't generate error message but we will loss the data.

## Implicit-typecasting:- (widening) or (up casting)

1. When we assign lower data type value to higher data type that typecasting is called up-casting.
2. When we perform up casting data no data loss.
3. It is also known as up-casting or widening.
4. Compiler is responsible to perform implicit typecasting.

## Explicit type-casting:- (Narrowing) or (down casting)

1. When we assign a higher data type value to lower data type that type-casting is called down casting.

2. When we perform down casting data will be loss.

3. It is also known as narrowing or down casting.

4. User is responsible to perform explicit typecasting.

**Note :- Parent class reference variable is able to hold child class object but Child class reference variable is unable to hold parent class object.**

```
class Parent {
};
class Child extends Parent {
};
Parent p = new Child();   //valid
Child c = new Parent(); //invalid
Example :-type casting
class Parent {
};
class Child extends Parent {
};
class Test  {
public static void main(String[] args)   {
//implicit typecasting (up casting)
byte b=120;
int i=b;   //[automatic conversion of byte-int]
System.out.println(b);
char ch='a';
int a=ch;  //[automatic conversion of char to int]
System.out.println(a);
long l1=20;
float f = l1; //[automatic conversion of long-float]
System.out.println(f);
}
}
```

## Technical Tasks

1. Write a program in Java with class Rectangle with the data fields width, length, area and color .The length, width and area are of double type and color is of string type .The methods are set_ length () , set_width (), set_ color(), and find_ area (). Create two object of Rectangle and compare their area and color. If area and color both are same for the objects then display "Matching Rectangles" otherwise display "Non matching Rectangle".
2. Write a program in Java to create a stack class with push() and pop methods. Create two objects of stack with 10 data item in both. Compare the top elements of both stack and print the comparison result.
3. Write Java program to show that private member of a super class cannot be accessed from derived classes.

4. Write a program to make a package Balance in which has Account class with Display_Balance method in it. Import Balance package in another program to access Display_Balance method of Account class.

## Exception Handling

## Information regarding Exception:-

- Dictionary meaning of the exception is abnormal termination.
- An expected event that disturbs or terminates normal flow of execution called exception.
- If the application contains exception then the program terminated abnormally the rest of the application is not executed.
- To overcome above limitation in order to execute the rest of the application must handle the exception. In java we are having two approaches to handle the exceptions.

**1) By using try-catch block.**

**2) By using throws keyword.**

## Exception Handling:-

- The main objective of exception handling is to get normal termination of the application in order to execute rest of the application code.
- Exception handling means just we are providing alternate code to continue the execution of remaining code and to get normal termination of the application.
- Every Exception is a predefined class present in different packages.
  **java.lang.ArithmeticException**
  **java.io.IOException**
  **java.sql.SQLException**
  **javax.servlet.ServletException**

The exception are occurred due to two reasons

**a. Developer mistakes**

**b. End-user mistakes.**

i. While providing inputs to the application.

ii. Whenever user is entered invalid data then Exception is occur.

iii. A file that needs to be opened can't found then Exception is occurred.

iv. Exception is occurred when the network has disconnected at the middle of the communication.

**Types of Exceptions:-** As per the sun micro systems standards The Exceptions are divided into three types

1) Checked Exception

2) Unchecked Exception

3) Error

## checked Exception:-

- ✓ The Exceptions which are checked by the compiler at the time of compilation is called Checked Exceptions. **IOException,SQLException,InterruptedException**……..etc.
- ✓ If the application contains checked exception the code is not compiled so must handle the checked Exception in two ways
  - By using try-catch block.
  - By using throws keyword.
- ✓ If the application contains checked Exception the compiler is able to check it and it will give intimation to developer regarding Exception in the form of compilation error.

## Unchecked Exception:-

- ✓ The exceptions which are not checked by the compiler at the time of compilation are called unchecked Exception. ArithmeticException,ArrayIndexOutOfBoundsException,NumberFormatException….etc
- ✓ If the application contains un-checked Exception code is compiled but at runtime JVM(Default Exception handler) display exception message then program terminated abnormally.
- ✓ To overcome runtime problem must handle the exception in two ways.
  - By using try-catch blocks.
  - By using throws keyword.

**Example :- different types of unchecked exceptions.**

```
class Test  {
public static void main(String[] args)   {
//java.lang.ArithmeticException: / by zero
System.out.println(10/0);
 //java.lang.ArrayIndexOutOfBoundsException
int[] a={10,20,30};   System.out.println(a[5]);
//java.lang.StringIndexOutOfBoundsException
System.out.println("ratan".charAt(10));
}
}
```

**Note-1:- If the application contains checked exception compiler generate information about exception so code is not compiled hence must handle that exception by using try-catch block or throws keyword it means for the checked exceptions try-catch blocks or throws keyword mandatory but if the application contains un-checked exception try-catch blocks or throws keyword is optional it means code is compiled but at runtime program is terminated abnormally.**

**Note 2:- In java whether it is a checked Exception or unchecked Exception must handle the Exception by using try-catch blocks or throws keyword to get normal termination of application.**

**Note-3:- In java whether it is checked Exception or unchecked exceptions are occurred at runtime but not compile time.**

## Error:-

- ✓ Errors are caused due to lack of system resources like, o Heap memory full. o Stack memory problem. o AWT component problems…..etc Ex: - StackOverFlowError, OutOfMemoryError, AssertionError……etc
- ✓ Exceptions are caused due to developers mistakes or end user supplied inputs but errors are caused due to lack of system resources.
- ✓ We are handle the exceptions by using try-catch blocks or throws keyword but we are unable to handle the errors.

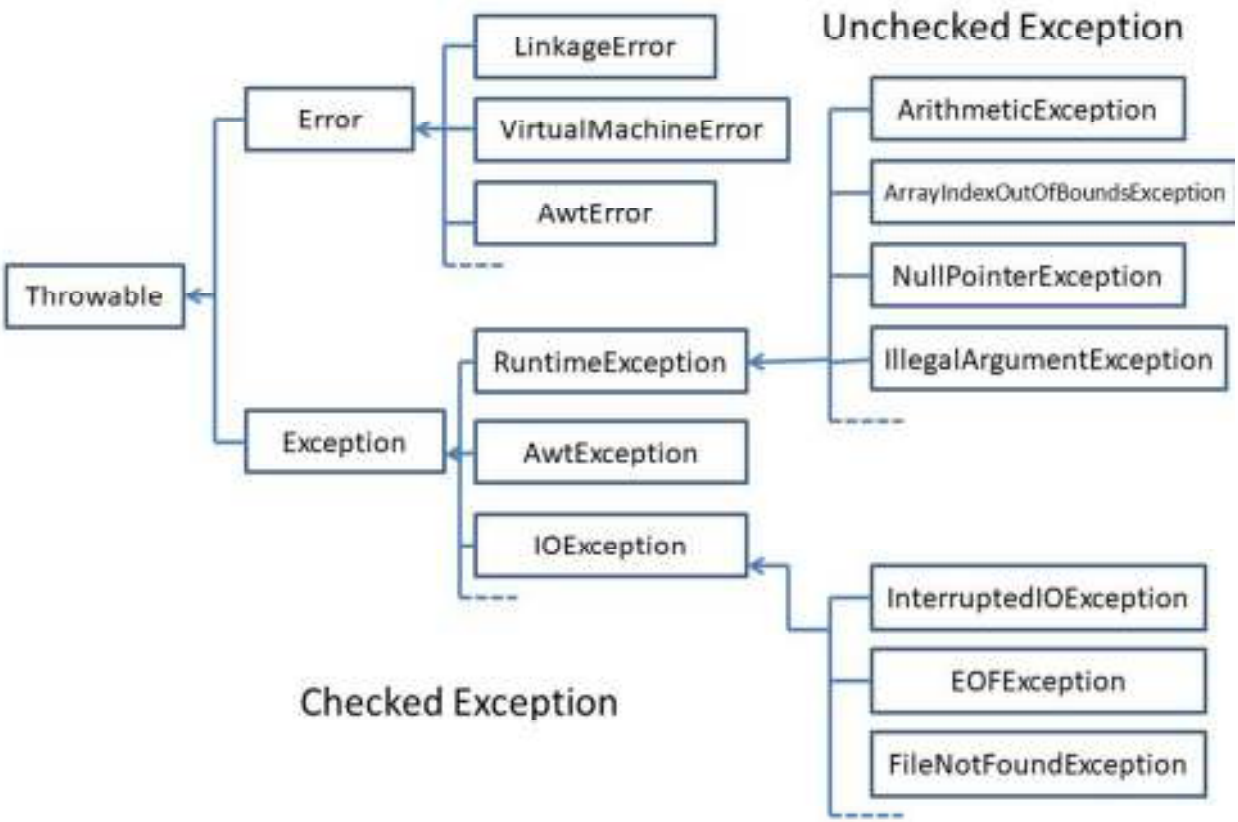
 

## Example:-

```
    class Test {
    public static void main(String[] args)  {
    Test[] t = new Test[100000000];
    }
};
```

**Exception in thread "main" java.lang.OutOfMemoryError: Java heap space**

## Exception Handling Tree Structure:-



In above tree Structure RuntimeException its child classes and Error its child classes are Unchecked remaining all exceptions are checked Exceptions.

## Exception handling key words:-

1) try

2) catch

3) finally

4) throw

5) throws

**Exception Handling:-** In java wether it is a checked Excepiton or unchecked Exception must handle the Exception by using try-catch blocks or throws to get normal termination of application.

## Exception handling by using Try –catch block:-

## Syntax:-

```
try {
exceptional code;
}
catch (ExceptionName  reference_variable) {
Code to run if an exception is raised;
}
```

## Example -1:- Application without try-catch

```
class Test  {
public static void main(String[] args)  {
System.out.println("Ravi 1st class");
System.out.println("Ravi 2nd class");
System.out.println("Ravi inter");
System.out.println("Ravi trainer");
System.out.println("Ravi weds Anushka"+(10/0));
System.out.println("Ravi's  kids");
}
}
```

D:\>java Test

Ravi 1st class

Ravi 2nd class

Ravi inter

Ravi trainer

**Exception in Thread "main" java.lang.ArithmeticException: / by zero**

## Application with try-catch blocks:-

Whenever the exception is raised in the try block JVM won't terminate the program immediately it will search corresponding catch block.

a. If the catch block is matched that will be executed then rest of the application executed and program is terminated normally.

b. If the catch block is not matched program is terminated abnormally.

```
class Test  {
public static void main(String[] args)  {
System.out.println("Ravi 1st class");
System.out.println("Ravi 2nd class");
System.out.println("Ravi inter");
System.out.println("Ravi trainer");
try   {
//Exceptional code
System.out.println("Ravi weds Anushka"+(10/0));
}
catch (ArithmeticException ae)   {
//alternate code
System.out.println("Ravi weds Aruna");
}
System.out.println("Ravi  kids");
}
}
```

D:\>java Test

Ravi 1st class

Ravi 2nd class

Ravi inter

Ravi trainer

Ravi weds Aruna

Ravi kids

**Example :- The way of handling the exception is varied from exception to the exception hence it is recommended to provide try with multiple number of catch blocks.**

```
import java.util.*;
class Test {
public static void main(String[] args) {
Scanner s=new Scanner(System.in); //Scanner object used to take dynamic input
System.out.println("provide the division value");
int n=s.nextInt();
try {
System.out.println(10/n);
String str=null;
System.out.println("u r name is :"+str);
System.out.println("u r name length is--->"+str.length());
}
catch (ArithmeticException ae) {
System.out.println("good boy zero not allowed geting Exception"+ae);
}
catch (NullPointerException ne) {
System.out.println("good girl getting Exception"+ne);
}
System.out.println("rest of the code");
}
}
```

**Output:- provide the division value: 5**

**Write the output**

**Output:- provide the division value: 0**

**Write the output**

## Finally block:-

1) Finally block is always executed irrespective of try and catch.

2) It is used to provide clean-up code

**a. Database connection closing. Connection.close();**

**b. streams closing. Scanner.close();**

**c. Object destruction . Test t = new Test();t=null;**

**It is not possible to write finally alone.**

a. try-catch-finally -→valid

b. try-catch -→ valid

c. catch-finally -→ invalid

d. try-catch-catch-finally -→ valid

e. try-finally -→ valid

f. catch-catch-finally -→ invalid

g. Try -→ invalid

h. Catch -→ invalid

i. Finally -→ invalid

**Syntax:-**

```
try {
risky code;
}
catch (Exception obj) {
handling code;
}
finally {
Clean-up code;(database connection closing,streams closing……etc)
}
```

**Example:-in only two cases finally block won't be executed**

**Case 1:-** whenever we are giving chance to try block then only finally block will be executed otherwise it is not executed.

```
class Test {
public static void main(String[] args)  {
System.out.println(10/0);
try   {
System.out.println("ratan");
}
finally   {
System.out.println("finally block");
}
System.out.println("rest of the code");
}
};
```

**D:\>java Test Exception in thread "main" java.lang.ArithmeticException: / by zero**

**at Test.main(Test.java:5)**

**Case 2:-In your program whenever we are using System.exit(0) the JVM will be shutdown hence the rest of the code won't be executed.**

```
class Test {
public static void main(String[] args) {
try {
System.out.println("Brijesh");
System.exit(0);
}
finally {
System.out.println("finally block");
}
System.out.println("rest of the code");
}
};
```

D:\>java Test

Brijesh

## Methods to print Exception information:-

```
class Test1 {
void m1() {
m2();
}
void m2() {
m3();
}
void m3() {
try{
System.out.println(10/0);
}
catch(ArithmeticException ae) {
System.out.println(ae.toString());
System.out.println(ae.getMessage());
ae.printStackTrace();
}
}
public static void main(String[] args) {
Test1 t = new Test1();
t.m1(); }};
```

**D:\DP>java Test1**

java.lang.ArithmeticException: / by zero **//toString() method output**

/ by zero **//getMessage() method output**

java.lang.ArithmeticException: / by zero **//printStackTrace() method**

at Test1.m3(Test1.java:8)

at Test1.m2(Test1.java:5)

at Test1.m1(Test1.java:3)

at Test1.main(Test1.java:17)

## Concept of Package:-

**java-language:-** in java James Gosling is maintained predefined support in the form of packages and these packages contains classes & interfaces, and these classes and interfaces contains predefined methods & variables.

Java-language-----→ packages-------→classes and interfaces--------→methods and variables

**Types of packages:-** There are two types of packages in java

1) Predefined packages.

2) User defined packages.

**Predefined packages:** The predefined packages are introduced by James Gosling and these packages contains predefined classes & interfaces and these class & interfaces contains predefined variables and methods. Example:- java.lang, java.io ,java.util…..etc

### User defined packages:-

- The packages which are defined by user, and these packages contains user defined classes and interfaces.
- Declare the package by using package keyword.
  **syntax : package package-name;**
  **example : package com.softpro;**
- Inside the source file it is possible to declare only one package statement and that statement must be first statement of the source file.

| **Example-1:valid**<br>package com.softpro;<br>import java.io.*;<br>import java.lang.*; | **Example-3:Invalid**<br>import java.io.*;<br>import java.lang.*;<br>package com.softpro; |
|---|---|
| **Example-2:Invalid**<br>import java.io.*;<br>package com.softpro;<br>import java.io.*; | **Example-4:Invalid**<br>package com.sofpro;<br>package com.slc; |

## Some predefined package and it's classes & interfaces:-

**Java.lang:-**The most commonly required classes and interfaces to write a sample program is encapsulated into a separate package is called java.lang package.

java

|-------→lang

|------→String (Class)

|------→StringBuffer (Class)

|-----→Object (Class)

|------→Runnable (Interface)

|-----→Clonable (Class)

Note:- the default package in the java programming is java.lang package.

**Java.io package:-**The classes which are used to perform the input output operations that are present in the java.io packages.

java

|-------→io

|------→ FileInputStream(class)

|------→ FileOutputStream(class)

|-----→ FileReader(class)

|------→ FileWriter(class)

|-----→ Serializable(interface)

**Java.net package:-**The classes which are required for connection establishment in the network that classes are present in the java.net package.

java

|-------→net

|------→ Socket(class)

|------→ ServerSocket(class)

|-----→ URL(class)

|------→ SocketOption(interface)

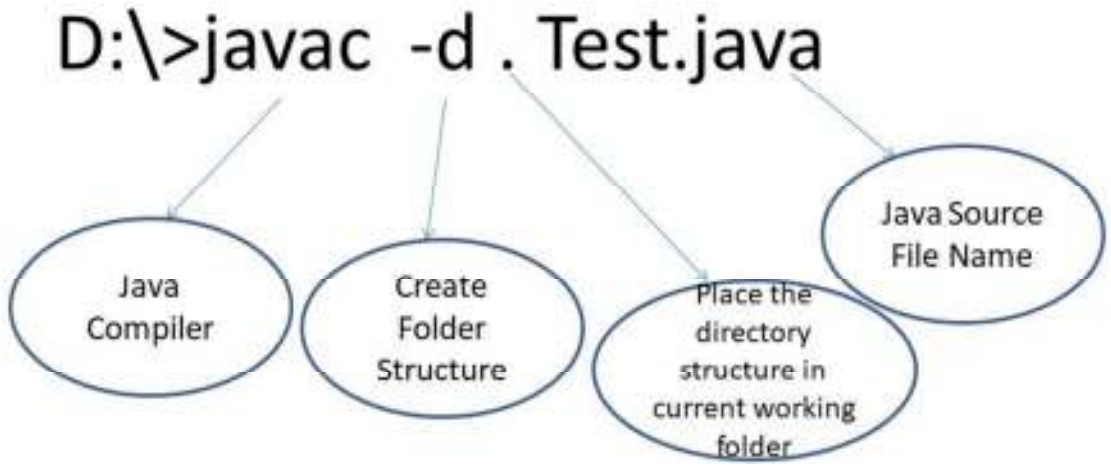
## Example-1:-

**Step-1: write the application with package statement.**

```
package com.softpro.java.corejava;
class Test  {
public static void main(String[] args)   {
System.out.println("package first example");
}
}
class A {
}
class B {
}
interface It {
}
```

**Step-2:** compilation process If the source file contains the package statement then compile that application by using fallowing command.




## Step-3:- folder Structure.

com

|--------→softpro

|------→java

|-----→corejava

|----→Test.class

|----→A.class

|-----→B.class

|----→it.class

### Step-4:-execution process.

Execute the .class file by using fully qualified name(class name with complete package structure)

java com.softpro.java.corejava.Test

**output : package first example**

# Interfaces:-

1. Interface is also one of the type of class it contains only abstract methods. And Interfaces not alternative for abstract class it is extension for abstract classes.

2. The abstract class contains atleast one abstract method but the interface contains only abstract methods.

3. For the interfaces the compiler will generates .class files.

4. Interfaces giving the information about the functionalities and these are not giving the information about internal implementation.

5. Inside the source file it is possible to declare any number of interfaces. And we are declaring the interfaces by using interface keyword.

## Syntax:-

**Interface interface-name**

**interface it1 { }** and bydefault above three methods are public the interface contains constants and these constants by default public static final

**Note-1 :- if u dont know the anything about implementation just we have the requirment specification them we should go for inteface**

**Note-2:- If u know the implementation but not completly then we shold go for abstract class**
**Note-3 :-if you know the implementation completly then we should go for concreate class**

**Both examples are same**

| | |
|---|---|
| interface it1<br>{<br>void m1();<br>void m2();<br>void m3();<br>} | abstract interface it1<br>{<br>public abstract void m1();<br>public abstract void m2();<br>public abstract void m3();<br>} |

**Note: - If we are declaring or not each and every interface method by default public abstract. And the interfaces are by default abstract hence for the interfaces object creation is not possible.**

## Example-1 :-

- Interface constrains abstract methods and by default these methods are “public abstract “.
- Interface contains abstract method for these abstract methods provide the implementation in the implementation classes.
- Implementation class is nothing but the class that implements particular interface.
- While providing implementation of interface methods that implementation methods must be public methods otherwise compiler generate error message “attempting to assign weaker access privileges”.

```
interface it1 // interface declaration
{
void m1();  //abstract method by default [public abstract]
void m2();//abstract method by default [public abstract]
void m3();//abstract method by default [public abstract]
}
class Test  implements it1  //Test is implementation class of It1 interface
{
public void m1()   //implementation method must be public
{
System.out.println(“m1-method implementation ”);
}
public void m2()
{
System.out.println(“m2-method implementation”);
}
public void m3()
{
System.out.println(“m3 –method implementation”);
}
public static void main(String[] args)
{
Test t=new Test();
t.m1();  t.m2();  t.m3();
}
}
```

## Example-2:-

- Interface contains abstract method for these abstract methods provide the implementation in the implementation class.
- If the implementation class is unable to provide the implementation of all abstract methods then declare implementation class with abstract modifier, take child class of implementation class then complete the implementation of remaining abstract methods.
- In java it is possible to take any number of child classes but at final complete the implementation of all abstract methods.

```
interface it1 // interface declaration
{
void m1();  //abstract method by default [public abstract]
void m2();//abstract method by default [public abstract]
void m3();//abstract method by default [public abstract]
}
//Test1 is abstract class contains 2 abstract methods m2() & m3()hence object creation not possible
abstract class Test1 implements it
{
public void m1()
{
System.out.println("m1 method");
}
};
//Test2is abstract class contains 1 abstract method m3()hence object creation not possible
abstract class Test2 extends Test1
{
public void m2() { System.out.println("m2 method");
}
};
//Test3 is normal class because it contains only normal methods hence object creation possible
class Test3 extends Test2
{
public void m3()
{
System.out.println("m3 method");
}
public static void  main(String[] args)
```

```
{
Test3 t = new Test3();
t.m1();  t.m2();  t.m3();
} };
```

## Technical Tasks

1. Write a program in Java to display name and roll number of students. Initialize respective array variables for 10 students. Handle ArrayIndexOutOfBoundsExeption, so that any such problem doesn't cause illegal termination of program.
2. Write a java program to facilitate user to handle any chance of divide by zero exception.
3. Write a program in java to create a String object. Initialize this object with your name. Find the length of your name using appropriate String method. Find whether character 'a' is in your name or not, if yes find the number of time 'a' it appear in your name. Print locations of occurrences of 'a' .Try same for different String objects.
4. Develop a program in Java to make reverse counter from 10 to 1.The numbers should be display in interval of 1 sec. like
   10
   9
   8
   .

# Multithreading:-

## Information about multithreading:-

1) The earlier days the computer's memory is occupied only one program after completion of one program it is possible to execute another program is called uni programming.

2) Whenever one program execution is completed then only second program execution will be started such type of execution is called co operative execution, this execution we are having lot of disadvantages.

a. Most of the times memory will be wasted.

b. CPU utilization will be reduced because only program allow executing at a time.

c. The program queue is developed on the basis co operative execution

**To overcome above problem a new programming style will be introduced is called multiprogramming.**

1) Multiprogramming means executing the more than one program at a time.

2) All these programs are controlled by the CPU scheduler.

3) CPU scheduler will allocate a particular time period for each and every program.

4) Executing several programs simultaneously is called multiprogramming.

5) In multiprogramming a program can be entered in different states.

a. Ready state.

b. Running state.

c. Waiting state.

6) Multiprogramming mainly focuses on the number of programs.

## Advantages of multiprogramming:-

1. The main advantage of multithreading is to provide simultaneous execution of two or more parts of a application to improve the CPU utilization.

2. CPU utilization will be increased.

3. Execution speed will be increased and response time will be decreased.

4. CPU resources are not wasted.

## Thread:-

1) Thread is nothing but separate path of sequential execution.

2) The independent execution technical name is called thread.

3) Whenever different parts of the program executed simultaneously that each and every part is called thread.

4) The thread is light weight process because whenever we are creating thread it is not occupying the separate memory it uses the same memory. Whenever the memory is shared means it is not consuming more memory.

5) Executing more than one thread a time is called multithreading.

## Information about main Thread:-

When a java program started one Thread is running immediately that thread is called main thread of your program.

1. It is used to create a new Thread(child Thread).

2. It must be the last thread to finish the execution because it perform various actions. It is possible to get the current thread reference by using currentThread() method it is a static public method present in Thread class.

```
class CurrentThreadDemo {
public static void main(String[] arhgs)  {
Thread t=Thread.currentThread();
System.out.println("current Thread--->"+t);   //change the name of the thread
t.setName("Ajay");
System.out.println("after name changed---> "+t);
}
};
```

## The main important application areas of the multithreading are

1. Developing video games

2. Implementing multimedia graphics.

3. Developing animations

## A thread can be created in two ways:-

1) By extending Thread class.

2) By implementing java.lang.Runnable interface

## First approach to create thread extending Thread class:-

**Step 1:- Our normal java class will become Thread class whenever we are extending predefined Thread class.**

```
class MyThread extends Thread {
};
```

**Step 2:- override the run() method to write the business logic of the Thread( run() method present in Thread class).**

```
class MyThread extends Thread {
public void run()  {
System.out.println("business logic of the thread");
System.out.println("body of the thread");
}
}
```

**Step 3:- Create userdefined Thread class object.**

```
MyThread t=new MyThread();
```

**Step 4:- Start the Thread by using start() method of Thread class.**

```
t.start();
```

## Example :-

```
class MyThread extends Thread //defining a Thread
{
//business logic of user defined  Thread
public void run()
{
for (int i=0;i<10;i++)
{
System.out.println("userdefined Thread");
}
}
};
class ThreadDemo {
public static void main(String[] args) //main thread started
{
MyThread t=new MyThread(); //MyThread is created
t.start(); //MyThread execution started
//business logic of main Thread
for (int i=0;i<10;i++)
{
System.out.println("Main Thread");
}
}
};
```

## Flow of execution:-

1) Whenever we are calling t.start() method then JVM will search start() method in the MyThread class since not available so JVM will execute parent class(Thread) start() method. Thread class start() method responsibilities

a. User defined thread is registered into Thread Scheduler then only decide new Thread is created.

b. The Thread class start() automatically calls run() to execute logics of userdefined Thread.

## Life cycle stages are:-

**1) New**

**2) Ready**

**3) Running state**

**4) Blocked / waiting / non-running mode**

**5) Dead state**

**New :-** MyThread t=new MyThread();

**Ready :-** t.start()

**Running state:-** If thread scheduler allocates CPU for particular thread. Thread goes to running state The Thread is running state means the run() is executed.

**Blocked State:-** If the running thread got interrupted of goes to sleeping state at that moment it goes to the blocked state.

**Dead State:-**If the business logic of the project is completed means run() over thread goes dead state.

## Second approach to create thread implementing Runnable interface:-

### Step 1:-our normal java class will become Thread class whenever we are implementing Runnable interface.

```
class MyClass implements Runnable {
};
```

### Step2*: override run method to write logic of Thread.*

```
class MyClass implements Runnable {
public void run()  {
System.out.println("Ajay from Softpro");
System.out.println("body of the thread");
}
}
```

### Step 3:- Creating a object.

MyClass  obj=new MyClass();

### Step 4:- Creates a Thread class object.

After new Thread is created it is not started running until we are calling start() method. So whenever we are calling start method that start() method call run() method then the new Thread execution started.

Thread t=new Thread(obj);

t.start();

## creation of Thread implementing Runnable interface :-

```
class MyThread implements Runnable {
public void run()  { //business logic of user defined  Thread
for (int i=0;i<10;i++)   {
System.out.println("userdefined Thread");
}
} .
};
class ThreadDemo {
public static void main(String[] args) //main thread started
{
 MyThread r=new MyThread(); //MyThread is created
Thread t=new Thread(r);
 t.start(); //MyThread execution started
//business logic of main Thread
for (int i=0;i<10;i++)
{
System.out.println("Main Thread");
}
}
};
```

### Creating two threads by extending Thread class using anonymous inner classes:-

```
class ThreadDemo {
public static void main(String[] args)  {
Thread t1 = new Thread() //anonymous inner class
{
public void run()
{
System.out.println("user Thread-1");
}
};
Thread t2 = new Thread() //anonymous inner class
{
public void run()
{
System.out.println("user thread-2");
}
};
t1.start();
t2.start();
}
};
```

## Technical Tasks

1. Write a java program to create five threads with different priorities. Send two threads of highest priority in sleep state. Check the aliveness of the threads and mark which dread is long listing.
2. Develop a program in Java to use **yield(), sleep()** and **stop()** methods to control the behavior of Thread.

# Collectionsframework (java.util)

## Pre-requisite topics for Collections framework:-

**1) AutoBoxing.**

**2) toString() method.**

**3) type-casting.**

**4) interfaces. .**

**5) for-each loop.**

**6) implementation classes.**

**7) compareTo() method.**

**8) Wrapper classes.**

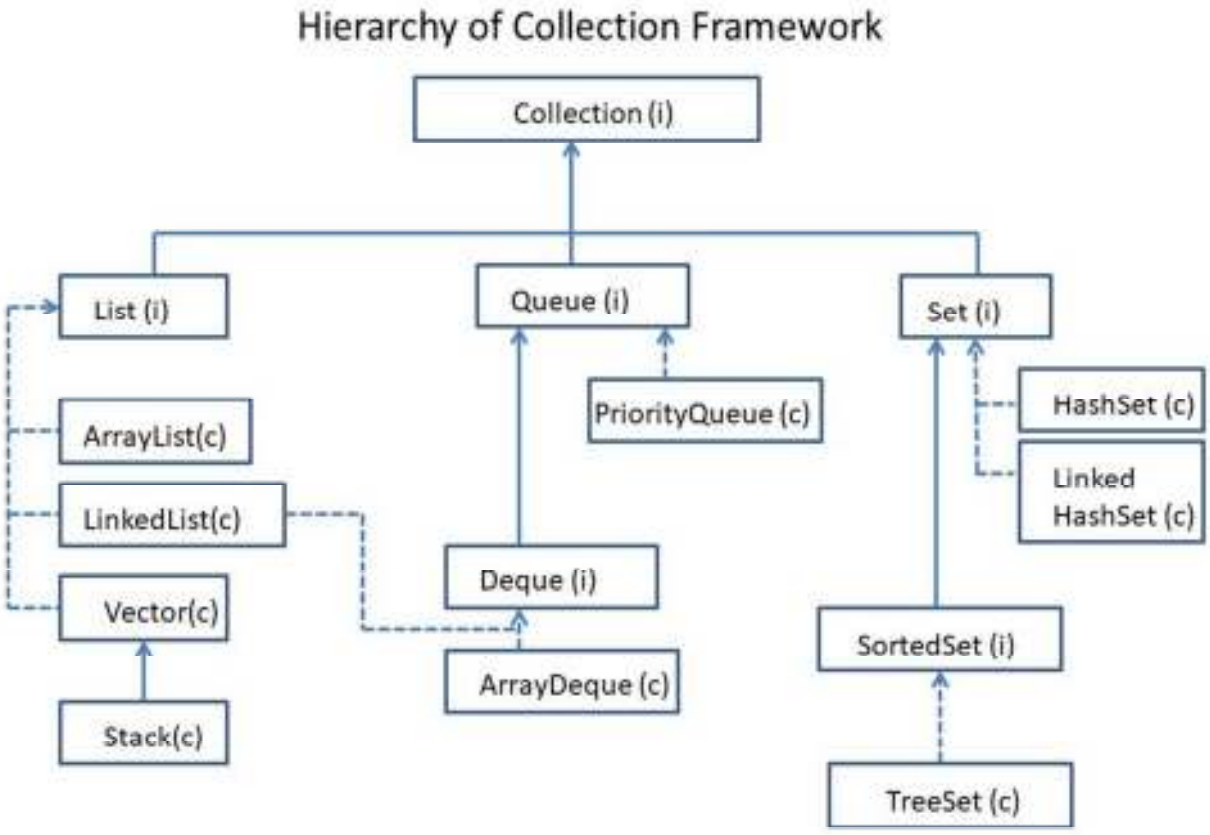
**9) Marker interfaces advantages.**

**10) Anonymous inner classes.**

- Collection framework contains group of classes and interfaces by using these classes & interfaces we are representing group of objects as a single entity.
- Collection is sometimes called a container. And it is object that groups multiple elements into a single unit.
- Collections are used to store, retrieve ,manipulate data.

**Note :-** The root interface of Collection framework is **Collection**.




## Characteristics of Collection frame work classes:-

1) The collect ion framework classes are introduced in different Versions.

2) Heterogeneous data allowed or not allowed.

All classes allowed heterogeneous data except two classes

i. TreeSet ii. TreeMap

3) Null insertion is possible or not possible.

4) Insertion order is preserved or not preserved.

Input --->e1 e2 e3 output --->e1 e2 e3 insertion order is preserved

Input --->e1 e2 e3 output --->e2 e1 e3 insertion order is not-preserved

5) Collection classes' methods are synchronized or non-synchronized.

6) Duplicate objects are allowed or not allowed.

add(e1)

add(e1)

7) Collections classes underlying data structures.

8) Collections classes supported cursors.

## Java.util.ArrayList:-

ArrayList is implementing List interface it widely used class in projects because it is providing functionality and flexibility To check parent class and interface use below command:-

**D:\>javap java.util.ArrayList**

public class java.util.ArrayList<E>

extends java.util.AbstractList<E>

implements java.util.List<E>,

java.util.RandomAccess,

java.lang.Cloneable,

java.io.Serializable

## ArrayList Characteristics:-

1) ArrayList Introduced in 1.2 version.

2) ArrayList stores Heterogeneous objects(different types).

3) Inside ArrayList we can insert Null objects.

4) ArrayList preserved Insertion order it means whatever the order we inserted the data in the same way output is printed.

a. Input ->e1 e2 e3 output ->e1 e2 e3 insertion order is preserved

b. Input →e1 e2 e3 output →e1 e3 e2 insertion order is not- preserved

5) ArrayList methods are non-synchronized methods.

6) Duplicate objects are allowed.

7) The under laying data structure is growable array.

8) By using cursor we are able to retrieve the data from ArrayList : Iterator , ListIterator

### Example:-Basic operations of ArrayList

```
import java.util.*;
class Test  {
public static void main(String[] args)   {
ArrayList al =new ArrayList();
al.add("A");
al.add("B");
al.add('a');
al.add(190);
al.add(null);
System.out.println(al);
System.out.println("ArrayList size-->"+al.size());
al.add(1,"A1");  //add the object at first index
System.out.println("after adding objects ArrayList size-->"+al.size());
System.out.println(al);
al.remove(1);  //remove the object index base
al.remove("A");  //remove the object on object base
System.out.println("after removeing elemetns arrayList size "+al.size());
System.out.println(al);
}
}
```

- In above example when we are adding primitive char value al.add('c') in ArrayList that value is automatically converted Character object format because ArrayList is able to store objects that is called AutoBoxing.
- When we print the ArrayList data for every object internally it is calling toString() method.

## Different ways to initialize values to ArrayList:-

**Case 1:initializing ArrayList by using asList()**

```
import java.util.*;
class ArrayListDemo  {
public static void main(String[] args)   {
ArrayList<String> al = new ArrayList<String>(Arrays.asList("Rohit","Yashi","Ajay"));
System.out.println(al);
}
}
```

**Case 2:- adding objects into ArrayList by using anonymous inner classes.**

```
import java.util.ArrayList;
class ArrayListDemo  {
public static void main(String[] args)   {
ArrayList<String> al = new ArrayList<String>()
{
{
add("Rohit");     add("Ajay");
}
};//semicolan is mandatory
System.out.println(al);
}
}
```

**Case 3:- normal approach to initialize the data**

```
import java.util.ArrayList;
class ArrayListDemo  {
public static void main(String[] args)
{
ArrayList<String> al = new ArrayList<String>();
al.add("Rohit");
al.add("Ajay");
System.out.println(al);
}
}
```

**Case 4:- ArrayList<Type> obj = new ArrayList<Type>(Collections.nCopies(count, object));**

```
import java.util.*;
class ArrayListDemo  {
public static void main(String[] args)   {
Emp e1 = new Emp(111,"Rohit");
ArrayList<Emp> al = new ArrayList<Emp>(Collections.nCopies(5,e1));
for (Emp e:al)
{
System.out.println(e.ename+"---"+e.eid);
}
}
}
```

**Case 5:-adding Objects into ArrayList by using addAll() method of Collections class.**

```
import java.util.*;
class Test  {
public static void main(String[] args)   {
ArrayList<String> al = new ArrayList<String>();
String[] strArray={"Rohit","Brijesh","Yashi"};
Collections.addAll(al,strArray);
System.out.println(al);
}
}
```

**Example :- retrieving data from generic version of ArrayList.**

```
import java.util.*;
class Emp
{
private int empid;
private String empname;
Emp(int empid,String empname)
{
this.empid=empid;
this.empname=empname;
}
}
```

```
class Test  {
public static void main(String[] args)   {
ArrayList<Emp> al = new ArrayList<Emp>();
al.add(new Emp(111,"Rohit"));
al.add(new Emp(222,"Yashi"));
al.add(new Emp(333,"Brijesh"));
for (Emp e : al)
{
System.out.println(e.eid+"---"+e.ename);
}
}
}
```

## we are able to retrieve objects  from collection classes in three ways:-

1) By using for-each loop.

2) By using cursors.

3) By using get() method.

**Cursors:-** Cursors are used to retrieve the Objects  from collection classes.

There are three types of cursors present in the java.

1. Enumaration

2. Iterator

3. ListIteator

## Applying Enumeration cursor on ArrayList:-

```
import java.util.*;
class ArrayListDemo  {
public static void main(String[] args)   {
ArrayList<String> al = new ArrayList<String>();
al.add("Ishan");
al.add("Rohit");
al.add("Yashi");
al.add("Ajay");
Enumeration<String> e = Collections.enumeration(al);
```

```
while (e.hasMoreElements())   {
String str = e.nextElement();
System.out.println(str);
}
}
}
```

## Sorting data by using sort() method of Collections class:-

we are able to sort ArrayList data by using sort()  method of Collections class and by default it perform ascending order .

If we want to perform descending order use **Collections.reverseOrder()**  method along with **Collection.sort()** method.

**Collections.sort(list ,Collections.reverseOrder());**

```
import java.util.*;
class Test  {
public static void main(String[] args) {
ArrayList<String> al = new ArrayList<String>();
al.add("Brijesh");
al.add("Yashi");
al.add("Ajay");
//printing ArrayList data
System.out.println("ArrayList data before sorting");
for (String str :al)   {
System.out.println(str);
}
//sorting ArrayList in ascending order
Collections.sort(al);
System.out.println("ArrayList data after sorting ascening order");
 for (String str1 :al)   {
System.out.println(str1);
}
//sorting ArrayList in decending order
Collections.sort(al,Collections.reverseOrder());
System.out.println("ArrayList data after sorting decening order");
for (String str2 :al)   {
System.out.println(str2);
}    } }
```

## Comparable vs Comparator :-

**Note :- it is possible to sort String and all wrapper objects because these objects are implementing Cloneable interface.**

- If we want to sort user defined class Emp based on eid or ename then your class must implements Comparable interface.
- Comparable present in java.lang package it contains only one method compareTo(obj) public abstract int compareTo(T);
- If your class is implementing Comparable interface then that objects are sorted automatically by using Collections.sort()And the objects are sorted by using compareTo() method of that class.
- By using comparable it is possible to sort the objects by using only one instance variable either eid or ename.

## Emp.java:-

```
class Emp implements Comparable  {
 int eid;
String ename;
Emp(int eid,String ename)  {
this.eid=eid;
this.ename=ename;
}
/* it is sorting the data by using enameinstance variable
public int compareTo(Object o)  {
Emp e = (Emp)o;
return  ename.compareTo(e.ename);
}  */  .
public int compareTo(Object o)  {
Emp e = (Emp)o;
if (eid == e.eid )   {
return 0;
}
else if (eid > e.eid)   {
return 1;
}
else    {
return -1;
}
}
}
```

### Test.java:-

```
import java.util.*;
class Test  {
public static void main(String[] args)   {
ArrayList al = new ArrayList();
al.add(new Emp(333,"Rohit"));
al.add(new Emp(222,"Yashi"));
al.add(new Emp(111,"Ajay"));
Collections.sort(al);
Iterator itr = al.iterator();
while (itr.hasNext())   {
Emp e = (Emp)itr.next();
System.out.println(e.eid+"---"+e.ename);
}
}
}
```

### Java.util.Comparator :-

- ✓ The class whose objects are stored do not implements this interface some third party class can also implements this interface.
- ✓ Comparable present in java.lang package but Comparator present in java.util package.

### Emp.java:-

```
class Emp  {
int eid;
String ename;
Emp(int eid,String ename)  {
this.eid=eid;   this.ename=ename;
}
}
```

**EidComp.java:-**

```
import java.util.Comparator;
class EidComp implements Comparator  {
public int compare(Object o1,Object o2)  {
Emp e1 = (Emp)o1;
Emp e2 = (Emp)o2;
if (e1.eid==e2.eid)   {
return 0;
}
else if (e1.eid>e2.eid)   {
return 1;
}
else   {
return -1;
}
}
}
```

**EnameComp.java:-**

```
import java.util.Comparator;
class EnameComp  implements Comparator  {
public int compare(Object o1,Object o2)  {
Emp e1 = (Emp)o1;
Emp e2 = (Emp)o2;
return (e1.ename).compareTo(e2.ename);
//return   -(e1.ename).compareTo(e2.ename);  //print data descending order
}
}
```

**Test.java:-**

```
import java.util.*;
class Test  {
public static void main(String[] args)  {
ArrayList<Emp> al = new ArrayList<Emp>();
al.add(new Emp(333,"Brijesh"));
al.add(new Emp(222,"Ajay"));
al.add(new Emp(111,"Rohit"));
```

```
al.add(new Emp(444,"Yashi"));
System.out.println("sorting by eid");
Collections.sort(al,new EidComp());
Iterator<Emp> itr = al.iterator();
while (itr.hasNext())   {
Emp e = itr.next();
System.out.println(e.eid+"---"+e.ename);
}
System.out.println("sorting by ename");
Collections.sort(al,new EnameComp());
Iterator<Emp> itr1 = al.iterator();
while (itr1.hasNext())   {
Emp e = itr1.next();
System.out.println(e.eid+"---"+e.ename);
}
}
}
```

## Iterator:-

1) Iterator cursor introduced in 1.2 versions.

2) Iterator  is a universal cursor applicable for all collection classes.

3) iterator() method is used to get Iterator object.

```
ex:- ArrayList al =new ArrayList();

al.add(10);

al.add(20);

al.add(30);

Iterator itr = al.iterator();
```

4) The Iterator object uses three methods to retrieve the objects from collections classes.

**public abstract boolean hasNext();**

This is used to check whether the Objects are available in collection class or not , if available returns true otherwise false.

**public abstract E next();**  This method used to retrieve the objects.

**public abstract void remove();** This method is used to remove the objects from collections classes.

5)

| **Normal version of Iterator** | **Generic version of ArrayList** |
|---|---|
| ArrayList al = new ArrayList();<br>al.add(10);<br>al.add(20);<br>al.add(30);<br>Iterator itr = al.iterator();<br>while (itr.hasNext()) {<br>Integer i = (Integer)itr.next();<br>**//normal version typecasting is required**<br>System.out.println(i);<br>} | ArrayList<Integer> al = new ArrayList<Integer>();<br>al.add(10);<br>al.add(20);<br>al.add(30);<br>Iterator<Integer> itr = al.iterator();<br>while (itr.hasNext()) {<br>Integer i = itr.next();<br>**//generic version type casting is not required**<br>System.out.println(i);<br>} |

6) By using Iterator cursor we are able to perform read and remove operations but it is not possible to perform update operation.

7) By using Iterator we are able to read the data only in forward direction.

## ListIterator:-

1. ListIterator cursor Introduced in 1.2 version

2. This cursor is applicable only for List type of classes(ArrayList,LinkedList,Vector,Stack...etc) hence it is not a universal cursor.

3. listIterator() method used to get ListIterator object

**ex:- LinkedList ll = new LinkedList();**

**ll.add(10);**

**ll.add(20);**

**ll.add(30);**

**ListIterator lstr = ll.listIterator();**

5. ListIterator contains fallowing methods
   **public abstract boolean hasNext();---->to check the Objects**
   **public abstract E next(); ----->to retrieve the objects top to bottom**

**public abstract boolean hasPrevious(); -→check the objects in previous direction**
**public abstract E previous(); ---->to retrieve the Objects from previous direction**
**public abstract int nextIndex();---->to get index**
**public abstract int previousIndex();--->to get the index from previous direction.**
**public abstract void remove(); --->to remove the Objects**
**public abstract void set(E); ----->to replace the particular Object**
**public abstract void add(E);---->to add new Objects**

5. By using this cursor we are able to read & remove & update the data.

6. By using this cursor we are able to read the data both in forward and backward direction.

## Application shows implementation class object of cursor interfaces (Enumeration , Iterator, ListIterator)

```
import java.util.*;
class Test {
public static void main(String[] args)  {
Vector v = new Vector();
v.addElement(10);
v.addElement(20);
v.addElement(30);
//it  returns implementation class object of Enumeration interface
Enumeration e = v.elements();
System.out.println(e.getClass().getName());
 //it  returns implementation class object of Iterator interface
Iterator itr = v.iterator();
System.out.println(itr.getClass().getName());
//it  returns implementation class object of ListIterator  interface
ListIterator lstr = v.listIterator();
System.out.println(lstr.getClass().getName());
}
};
```

# Technical Tasks

1. Write a program to iterate through an ArrayList.
   **Hint:** All of the collection classes provides iterator() method to iterate through the collection. The iterator() method returns the Iterator object through which you can access the collection elements in an order.
2. Develop a program in java to sort an ArrayList using Comparator.
3. Develop a program in java to store the five elements in ArrayList and reverse those elements.
4. Develop a program in java to take a ArrayList store five employee names in this ArrayList display the names using Iterator.

## LinkedList:-

**Class LinkedList extends AbstractSequentialList implements List,Deque,Queue**

1) Introduced in 1.2 v

2) Heterogeneous objects are allowed.

3) Null insertion is possible.

4) Insertion ode is preserved

5) LinkedList methods are non-synchronized method

6) Duplicate objects are allowed.

7) The under laying data structure is double linkedlist.

8) cursors :- Iterator,ListIterator Ex:-LinkedList with generics.

## Ex:-

```
import java.util.*;
class Test {
public static void main(String[] args)  {
LinkedList<String> l=new LinkedList<String>();
l.add("B");
l.add("C");
l.add("D");
l.add("E");
l.addLast("Z");//it add object in last position
l.addFirst("A");//it add object in first position
l.add(1,"A1");//add the Object spcified index
System.out.println("original content:-"+l);
```

```
l.removeFirst();  //remove first  Object
l.removeLast();  //remove last t Object
System.out.println("after deletion first & last:-"+l);
l.remove("E");  //remove specified Object
l.remove(2);  //remove the object of specified index
System.out.println("after deletion :-"+l);//A1 B D
String val = l.get(0);  //get method used to get t0he element
l.set(2,val+"cahged");//set method used to replacement
System.out.println("after seting:-"+l);
} };
```

## HashSet:-

1) Introduced in 1.2 v.

2) Duplicate objects are not allowed if we are trying to insert duplicate values then we won't get any compilation errors an won't get any Execution errors simply add method return old value.

3) Null insertion is possible but if we are inserting more than one null it return only one null value.

4) Heterogeneous objects are allowed.

5) The under laying data structure is HashTable.

6) It is not maintain any order the elements are return in any random order .[Insertion order is not preserved].

7) Methods are non-synchronized.

8) cursor  : Iterator

## Example:-

```
import java.util.*;
class  Test {
public static void main(String[] args)   {
//HashSet   object creation
HashSet<String> h = new HashSet<String>();
 h.add("A");
h.add("B");
```

```
h.add("C");
h.add("D");
h.add("D");   //creation of Iterator Object
Iterator<String> itr = h.iterator();
 while (itr.hasNext())   {
String str  = itr.next();
System.out.println(str);
}
}
}
```

## Example Application:-

```
import java.util.*;
 class Test  {
public static void main(String[] args)   {
HashSet<String> h = new HashSet<String>();
h.add("Ajay");
h.add("Rohit");
h.add("Brijesh");
HashSet<String> hsub = new HashSet<String>();
hsub.add("no1");
hsub.add("no2");
hsub.addAll(h);
System.out.println(hsub.contains("Rohit"));
hsub.remove("Brijesh");
System.out.println(hsub.containsAll(h));
System.out.println(hsub);
hsub.removeAll(h);
System.out.println(hsub);
hsub.retainAll(h);
System.out.println(hsub);
}
}
```

## LinkedHashSet:-

1. Introduced in 1.4 version and It is a child class of HashSet.

2. Duplicate objects are not allowed if we are trying to insert duplicate values then we won't get any compilation errors an won't get any Execution errors simply add method return false.

3. Null insertion is possible.

4. Heterogeneous objects are allowed

5. The under laying data structure is LinkedList & hashTable.

6. Insertion order is preserved.

7. Methods are non-synchronized.

8. Cursors :- Iterator

## Example:-

```
import java.util.*;
class  Test {
public static void main(String[] args)   {
Set<String> h = new LinkedHashSet<String>();
h.add("A");
h.add("B");
h.add("C");
h.add("D");
h.add("D");  //retrieving objects by using Iterator cursor
Iterator<String> itr = h.iterator();
while (itr.hasNext())   {
String str  = itr.next();
System.out.println(str);
}
//retrieving objects by using Enumeration cursor
Enumeration<String> e = Collections.enumeration(h);
while (e.hasMoreElements())   {
System.out.println(e.nextElement());
}
}
}
```

## TreeSet:-

1. TreeSet is same as HashSet but TreeSet sorts the elements is ascending order but HashSet does not maintain any order.

2. The underlying data Structure is BalencedTree.

3. Insertion order is not preserved it is based some sorting order.

4. Heterogeneous data is not allowed.

5. Duplicate objects are not allowed

6. Null insertion is possible only once.

## Example Application:-

```
import java.util.*;
class Test  {
public static void main(String[] args)   {
TreeSet<String> t = new TreeSet<String>();
t.add("Ajay");
t.add("Rohit");   t.add("Yashi");
System.out.println(t);
TreeSet<Integer> t1 = new TreeSet<Integer>();
t1.add(10);
t1.add(12);
t1.add(8);
System.out.println(t1);
}
}
```

## Map:-

1. Map is a child interface of collection.

2. Up to know we are working with single object and single value where as in the map collections we are working with two objects and two elements.

3. The main purpose of the collection is to compare the key value pairs and to perform necessary operation.

4. The key and value pairs we can call it as map Entry.

5. Both keys and values are objects only.

6. In entire collection keys can’t be duplicated but values can be duplicate.

## HashMap:-

1) interdicted in 1.2 version

2) Heterogeneous data allowed.

3) Underlying data Structure is HashTable.

4) Duplicate keys are not allowed but values can be duplicated.

5) Insertion order is not preserved.

6) Null is allowed for key(only once)and allows for values any number of times.

7) Every method is non-synchronized so multiple Threads are operate at a time hence permanence is high.

8) cursor :- Iterator.

## Example :-

```
import java.util.*;
class Test {
public static void main(String[] args)  {
//creation of HashMap Object
HashMap h = newHashMap();
h.put("Brijesh",111);     //h.put(key,value);
h.put("Yashi",222);
h.put("Rohit",333);
Set s1=h.keySet();  //used to get all keys
System.out.println("all keys:--->"+s1);
Collection c = h.values();  //used to get all values
System.out.println("all values--->"+c);
Set ss = h.enrySet();  //it returns all entryes nathing but [key,value]
System.out.println("all entries--->"+ss); //get the Iterator Object
Iterator itr = ss.iterator();
while (itr.hasNext()) {
//next() method retrun first entry to represent that entery do typeCasting
Map.Entry  m= (Map.Entry)itr.next();
System.out.println(m.getKey()+"----"+m.getValue()); //printing key and value
}
}
};
```

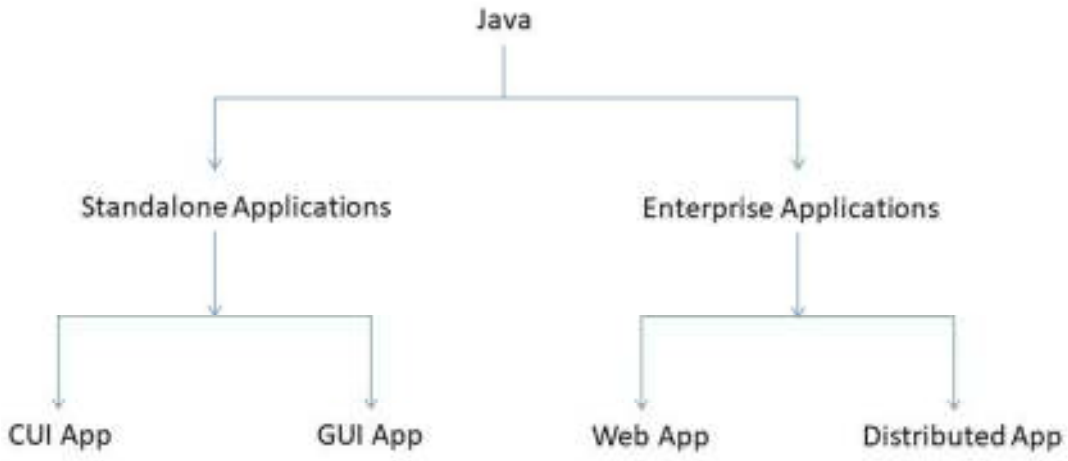
## Technical Tasks

1. Develop a program in java to sort a LinkedList using Comparator.
2. Develop a program in java to take a LinkedList store five employee names in this LinkedList display the names using Iterator.
3. Develop a program in java to store the five elements in LinkedList and reverse those elements.
4. Write a program in java to make a HashTable add few items into the HashTable. Now search an item into the HashTable.
5. Write a program in java to remove duplicate entries in an array.

# Concept of Servlet

In general by using Java technology we are able to design following 2 types of applications.




## 1. Standalone Applications:

These are the java applications, which will be designed without using Client-Server Architecture.

There are 2 types of Standalone applications.

1. CUI Applications
2. GUI Applications

### CUI Applications:-

CUI Applications are the standalone applications, which will be designed in such a way to take input and to provide output by using Command prompt, where Command prompt is acting as user interface and it able to support character data. So that the java application which will design on the basis of command prompt i.e. character user interface is called as CUI application.

### GUI Applications:-

GUI Applications are the standalone applications, which will be designed in such a way to take input and to provide output by using a collection of Graphic component, where the collection of Graphic components is acting as an user interface and it able to support GUI component so that

the java application which will be design on the basis of Graphical user interface is called as GUI Application.

## 2. Enterprise Applications:

These are the java applications, which will be designed on the basis of Client-Server Architecture. There are 2 types of Enterprise applications.

1. Web Applications

2. Distributed Applications

**Web Applications:-** Web Application is a collection of web components, it will be executed by using web containers like Servlet Container to execute servlets, Jsp Container to execute JSP's and so on.

**Distributed Applications:-** Distributed Application is a collection of distributed components, it will be executed by using the distributed containers like EJB's Container to execute EJB's.

At the beginning stages of the computer we have Client-Server architecture, where the purpose of server is to hold up some resources and to share that resources to all the clients as per the client request.

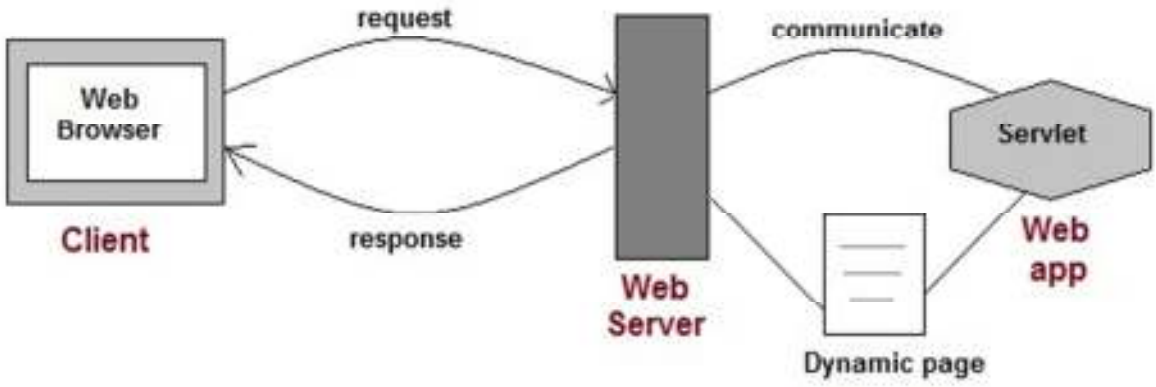
In the above context, when we send a request from client to server for a particular resource then server will identify requested resource, pick up the content and send that content as response to client without performing any particular action at server. In this case the response which was generated by server is called as static response.

## Introduction to Servlet

**Servlet** Technology is used to create web applications. **Servlet** technology uses Java language to create web applications.

Web applications are helper applications that resides at web server and build dynamic web pages. A dynamic page could be anything like a page that randomly chooses picture to display or even a page that displays the current time.



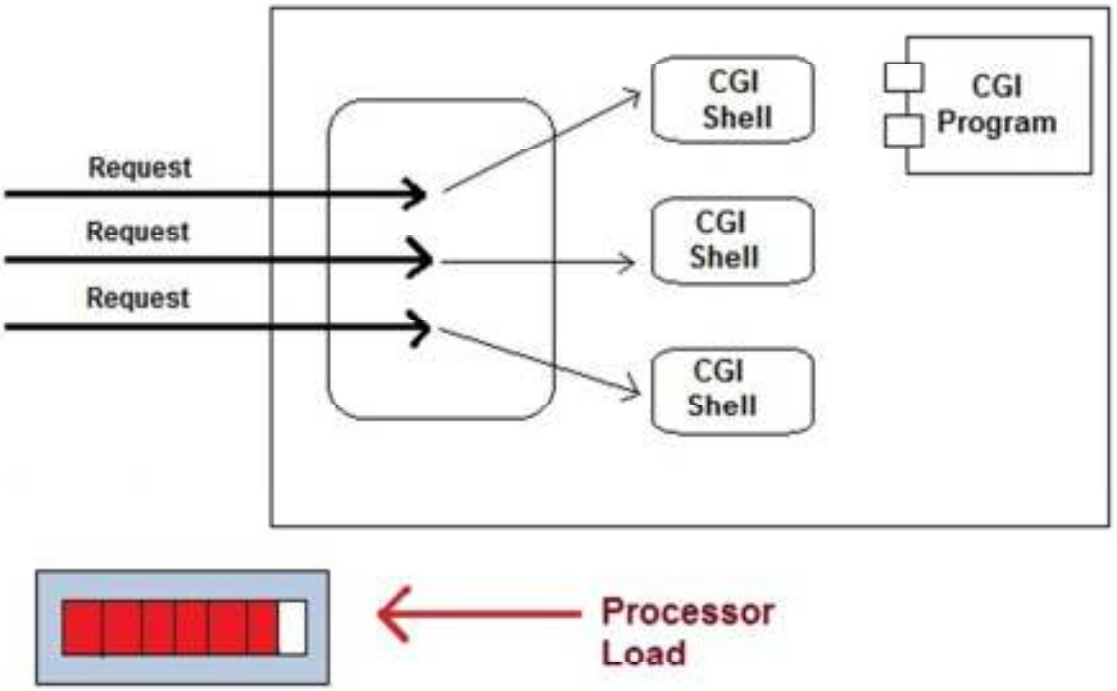

As Servlet Technology uses Java, web applications made using Servlet are **Secured**, **Scalable** and **Robust**

## CGI (Common Gateway Interface):-

Before Servlets, CGI(Common Gateway Interface) programming was used to create web applications. Here's how a CGI program works :

- User clicks a link that has URL to a dynamic page instead of a static page.
- The URL decides which CGI program to execute.
- Web Servers run the CGI program in seperate OS shell. The shell includes OS enviroment and the process to execute code of the CGI program.
- The CGI response is sent back to the Web Server, which wraps the response in an HTTP response and send it back to the web browser.




## Drawbacks of CGI programs:-

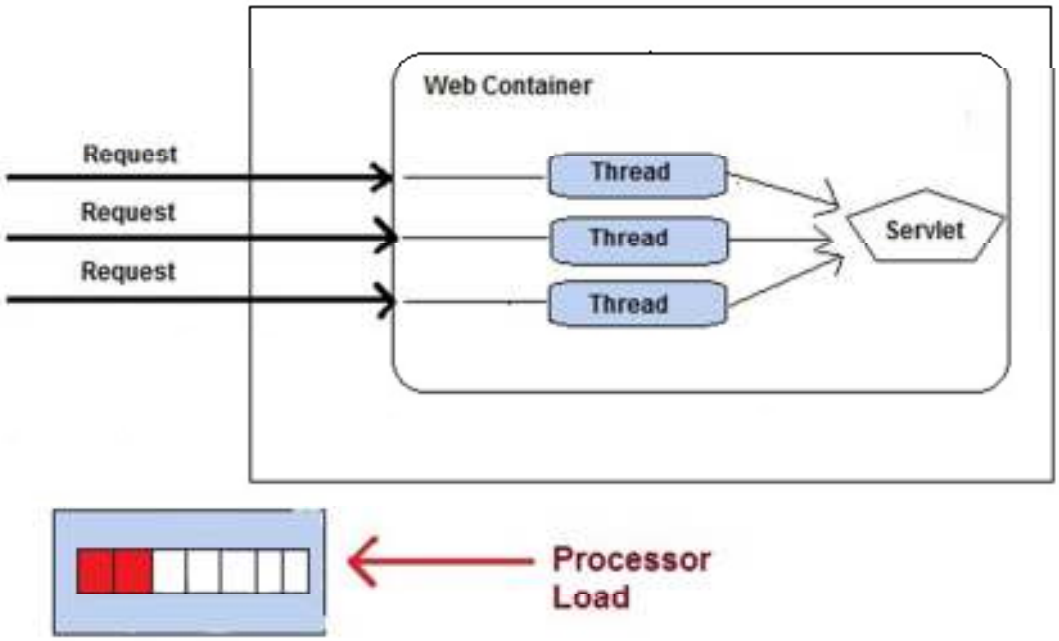
- High resposne time because CGI programs execute in their own OS shell.
- CGI is not scalable.
- CGI programs are not always secure or object-oriented.
- It is Platform dependent.

Because of these disadvantages, developers started looking for better CGI solutions. And then Sun Microsystems developed **Servlet** as a solution over traditional CGI technology.

## Advantages of using Servlets:-

- Less response time because each request runs in a separate thread.
- Servlets are scalable.
- Servlets are robust and object oriented.
- Servlets are platform independent.




## Servlet API

Servlet API consists of two important packages that encapsulates all the important classes and interface, namely :

- **javax.servlet**
- **javax.servlet.http**

## How a Servlet Application works

**Web container** is responsible for managing execution of servlets and JSP pages for Java EE application.

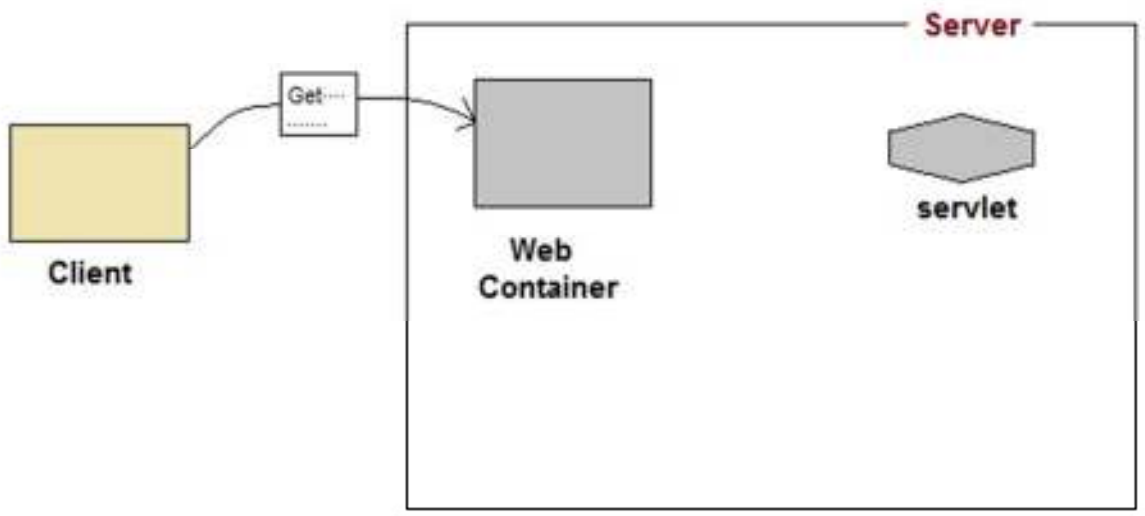
When a request comes in for a servlet, the server hands the request to the Web Container. **Web Container** is responsible for instantiating the servlet or creating a new thread to handle the request. Its the job of Web Container to get the request and response to the servlet. The container creates multiple threads to process multiple requests to a single servlet.

**Servlets don't have a main() method**. Web Container manages the life cycle of a Servlet instance.

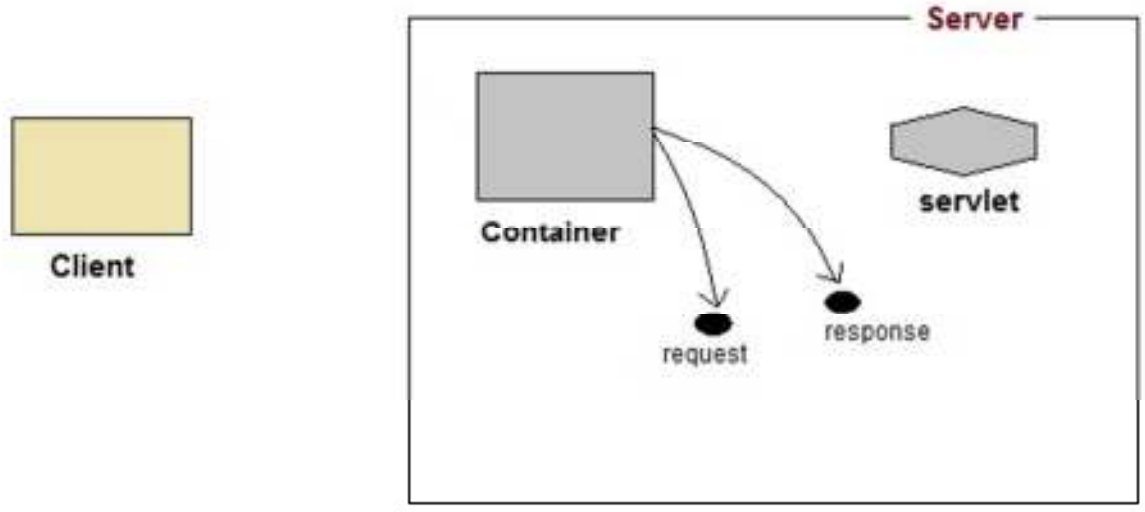
## Quick Revision on How a Servlet works:-

User sends request for a servlet by clicking a link that has URL to a servlet.



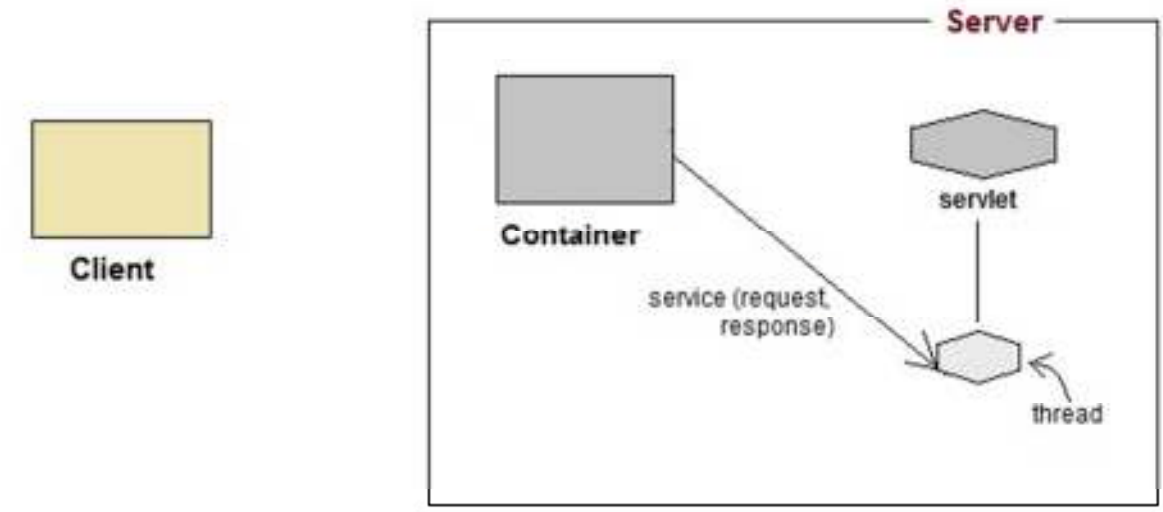

2. The container finds the servlet using **deployment descriptor** and creates two objects :

a. **HttpServletRequest**
b. **HttpServletResponse**

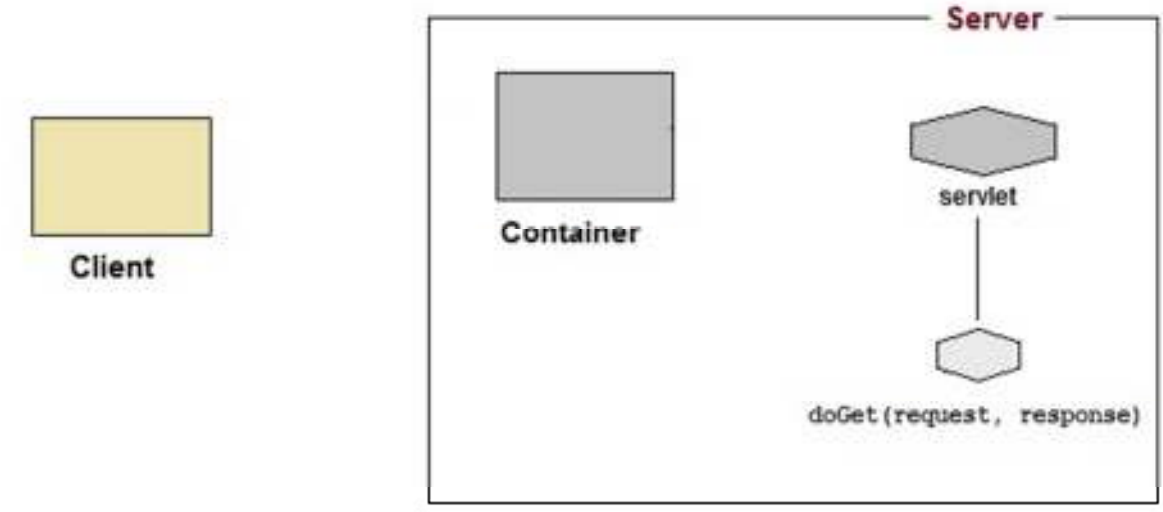


3. Then the container creates or allocates a thread for that request and calls the Servlet's service()method and passes the **request, response** objects as arguments.

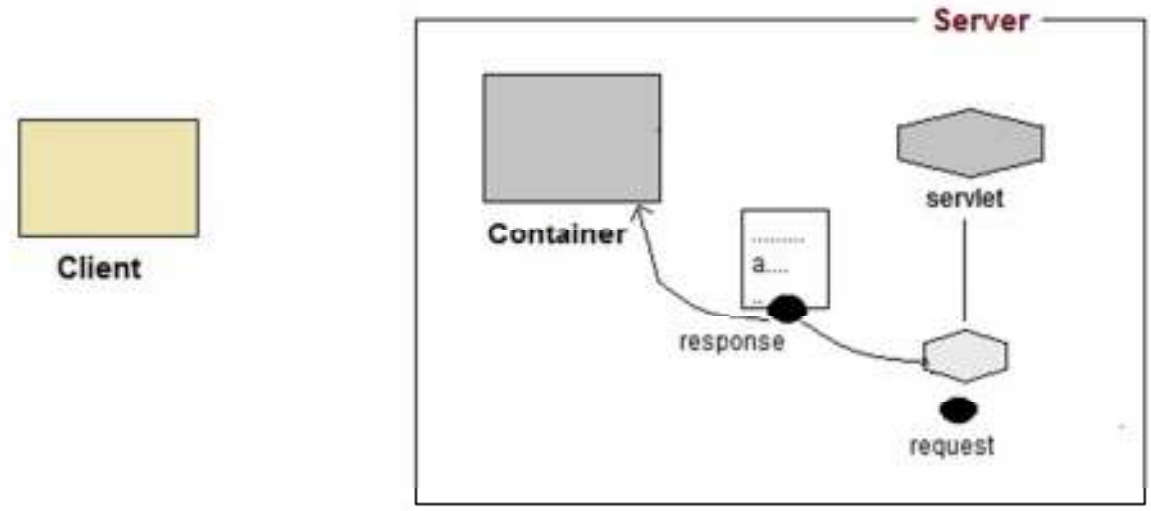



4. The service() method, then decides which servlet method, doGet() or doPost() to call, based on **HTTP Request Method**(Get, Post etc) sent by the client. Suppose the client sent an HTTP GET request, so the service() will call Servlet's doGet() method.

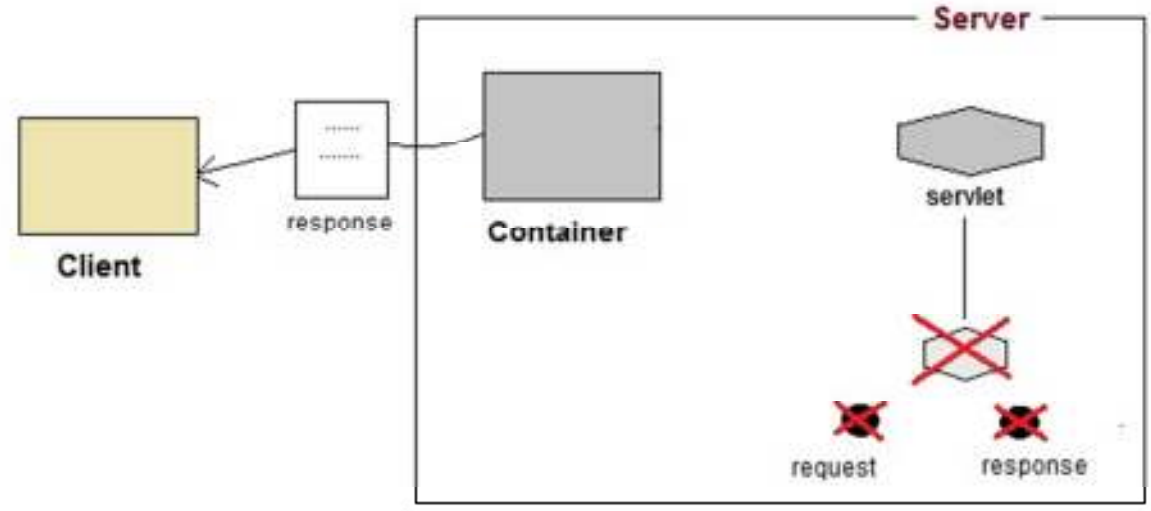


5. Then the Servlet uses response object to write the response back to the client.

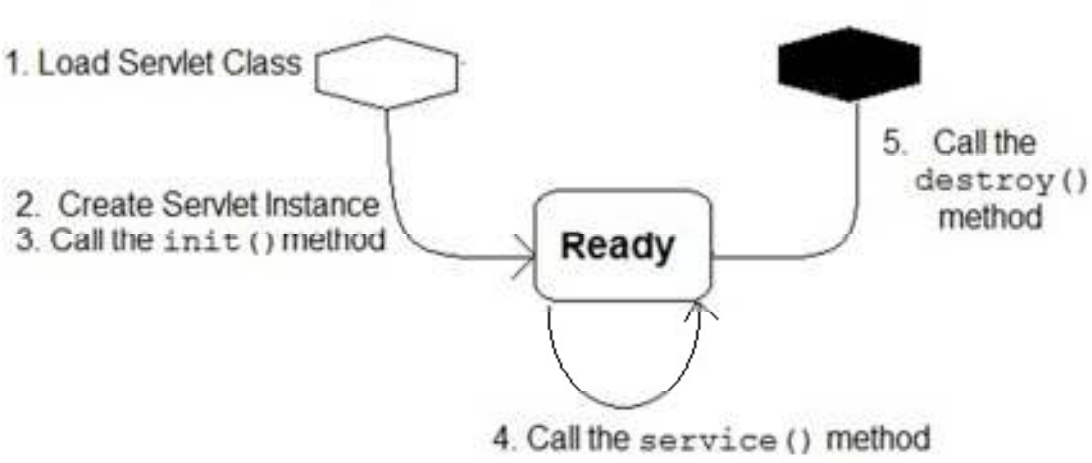



6. After the service() method is completed the **thread** dies. And the request and response objects are ready for **garbage collection**.



## Servlet Life Cycle




1. **Loading Servlet Class :** A Servlet class is loaded when first request for the servlet is received by the Web Container.
2. **Servlet instance creation :**After the Servlet class is loaded, Web Container creates the instance of it. Servlet instance is created only once in the life cycle.
3. **Call to the init() method :** init() method is called by the Web Container on servlet instance to initialize the servlet.

**Signature of init() method :**

public void init(ServletConfig config) throws ServletException

4. **Call to the service() method :** The containers call the service() method each time the request for servlet is received. The service() method will then call the doGet() or doPost() methos based ont eh type of the HTTP request.
   **Signature of service() method :**
   public void service(ServletRequest request, ServletResponse response) throws ServletException, IOException
5. **Call to destroy() method:** The Web Container call the destroy() method before removing servlet instance, giving it a chance for cleanup activity.

## Marriage Eligibility Test Application using Servlet:-

### index.html page code:-

```
<html>
  <head>
    <title>MET</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body bgcolor="aqua">
    <form action="test" method="post">
      <h1>Marriage Eligibility Test</h1>
      <hr/>
      Enter your name
      <input type="text" name="name" placeholder="Enter Name"/><br/><br/>
      Enter your age
      <input type="number" name="age" placeholder="Enter Age"/><br/><br/>
      Select your gender
      <input type="radio" name="gender" value="Male"/>Male
      <input type="radio" name="gender" value="Female"/>Female
      <br/><br/>
      <input type="submit" value="Test"/>
    </form>
  </body>
</html>
```

## Marriage Eligibility Test

Enter your name

Enter your age

Select your gender Male Female

Test

### Test.java Code:-

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(urlPatterns = {"/test"})
public class test extends HttpServlet {
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet test</title>");
        out.println("</head>");
        out.println("<body>");
        String name=request.getParameter("name");
        int age=Integer.parseInt(request.getParameter("age"));
        String gender=request.getParameter("gender");
        if(age>=21 && gender.equals("Male"))
        {
        out.println("<h1>Congo! "+name+" you are eligible for marriage</h1>");
        }
```

```
        else if(age>=18 && gender.equals("Female"))
        {
        out.println("<h1>Congo! "+name+" you are eligible for marriage</h1>");
        }
        else
        {
        out.println("<h1>Sorry! "+name+" wait for suitable time.</h1>");
        }
        out.println("</body>");
        out.println("</html>");
    }
}
 protected void doGet(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
        processRequest(request, response);
    }
 protected void doPost(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
        processRequest(request, response);
    }
```

## Simple Calculator Application using Servlet:-

## index.html code:-

```
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <form action="Calc" method="post">
            <h2 style="color:blue">Simple Calculator</h2>
            <hr>
            Enter first number
            <input type="text" name="num1"/><br><br>
            Enter second number
            <input type="text" name="num2"/><br><br>
```

```
            Select operator
            <input type="radio" name="op" value="add"/>+
            <input type="radio" name="op" value="sub"/>-
            <input type="radio" name="op" value="mult"/>*
            <input type="radio" name="op" value="div"/>/
            <input type="radio" name="op" value="mod"/>%
            <br><br>
            <input type="submit" value="Submit"/>
        </form>
    </body>
</html>
```

## Simple Calculator

---

Enter first number

Enter second number

Select operator + - * / %

Submit

---

### Calc.java code:-

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(urlPatterns = {"/Calc"})
public class Calc extends HttpServlet {
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
```

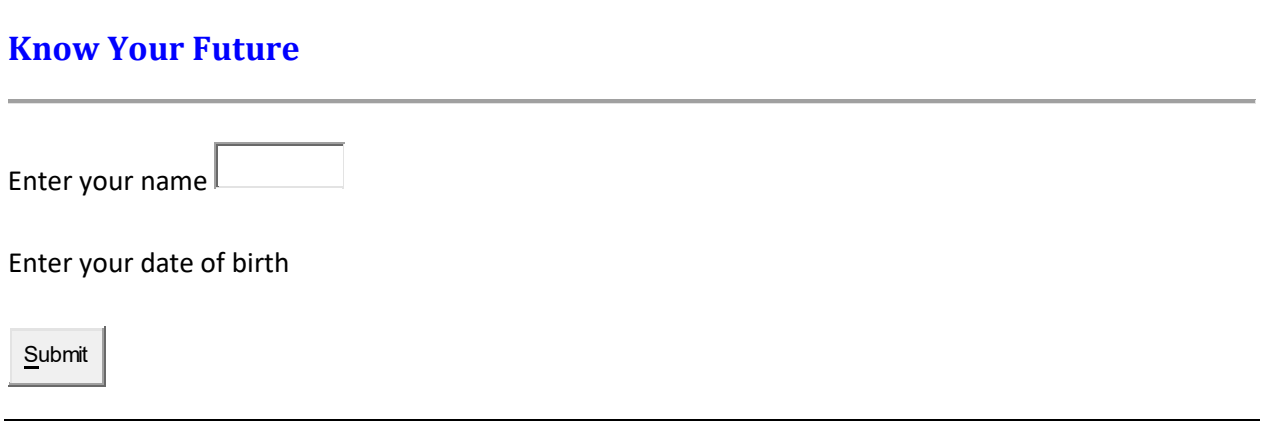
```
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet Calc</title>");
            out.println("</head>");
            out.println("<body>");
            int n1=Integer.parseInt(request.getParameter("num1"));
            int n2=Integer.parseInt(request.getParameter("num2"));
            String op=request.getParameter("op");
            int res=0;
            if(op.equals("add"))
                res=n1+n2;
            else if(op.equals("sub"))
                res=n1-n2;
            else if(op.equals("mult"))
                res=n1*n2;
            else if(op.equals("div"))
                res=n1/n2;
            else if(op.equals("mod"))
                res=n1%n2;
            out.println("<h2>Result : "+res+"</h2>");
            out.println("</body>");
            out.println("</html>");
        }
    }
protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
}
```

## Future Prediction Application using Servlet:-

## index.html code:-

```
<html>
  <head>
    <title>TODO supply a title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <form action="Future" method="post">
      <h2 style="color:blue">Know Your Future</h2>
      <hr>
      Enter your name
      <input type="text" name="name"/><br><br>
      Enter your date of birth
      <input type="date" name="dob"/><br><br>
      <input type="submit" value="Submit"/>
    </form>
  </body>
</html>
```



## Future.java code

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(urlPatterns = {"/Future"})
public class Future extends HttpServlet {
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet Future</title>");
        out.println("</head>");
        out.println("<body>");
        String name=request.getParameter("name");
        String dob=request.getParameter("dob");
        String str=dob.replace("-", "0");
        out.println("<h2>Hello! "+name+"</h2>");
        int n=0;
        for(int i=0;i<str.length();i++)
        {
        n=n+Integer.parseInt(str.charAt(i)+"");
        }
        while(n>9)
        {
        n=n/10+n%10;
        }
        out.println("<h2>Your Lucky No.="+n+"</h2>");
        String future="";
        switch(n)
        {
           case 1:
              future="You are an intelligent person";
```

```
            break;
          case 2:
            future="You can be a successfull software developer";
            break;
          case 3:
            future="You are a delligent person";
            break;
          case 4:
            future="You can get marry soon";
            break;
          case 5:
            future="You can be primeminister of India";
            break;
          case 6:
            future="You are an average person";
            break;
          case 7:
            future="Your future is very bright";
            break;
          case 8:
            future="You can be a successfull scientist";
            break;
          case 9:
            future="All is well in your life";
            break;
          default:
            future="Are you came from another planet?";
            break;
        }
        out.println("<h2>Your future is : "+future+"</h2>");
        out.println("</body>");
        out.println("</html>");
      }
    }
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
      processRequest(request, response);
    }
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
      processRequest(request, response);
    }
```

# JDBC (Java Database Connectivity)

## Storage Areas

As part of the Enterprise Application development it is essential to manage the organizations data like Employee Details, CustomerDetails, Products Details..etc

-->To manage the above specified data in enterprise applications we have to use storage areas (Memoryelements).There are two types of Storage areas.

1) Temporary Storage Areas:

These are the memory elements, which will store the data temporarily Eg:Buffers,Java Objects

2) Permanent Storage Objects:

These are the memory elements which will store data permanently.
Eg:FileSystems,DBMS,DataWareHouses.

**File Systems:** It is a System, it will be provided by the local operating System.

--->Due to the above reason File Systems are not suitable for platform independent technologies like JAVA.

--->File Systems are able to store less volumes of the data.

--->File Systems are able to provide less security.

--->File Systems may increases data Redundancy.

--->In case of File Systems Query Language support is not available. So that all the database operations are complex.

## DBMS:-

- Database Management System is very good compare to file System but still it able to store less data when compared to DataWareHouses.
- DBMS is very good at the time of storing the data but which is not having Fast Retrieval Mechanisms.

## There are three types of DBMS:-

1. RDMS(Relational Database Management Systems)
2. OODBMS(Object Oriented DataBase Management Systems)
3. ORDBMS(Object Relational DataBase Management Systems)

### 1. Relational Database Management Systems:

It is a DBMS,it can be used to represent the data in the form of tables.
This DBMS will use SQL3 as a Query Language to perform DataBase Operations.

### 2. Object Oriented Database Management System:

This database management system will require OQL(Object Query Language)as Query language to perform database operations.

### 3. Object Relational Database Management System:

- It is a Database Management System, it will represents some part of data in the form of Tables and some other part of the data in the form of objects
- This Database Management System will require SQl3 as Query Language to perform database operations.

## Query Proccessing System:-

1. Query Tokenization
2. Query Processing
3. Query Optimization
4. Query Execution

## Query Processing System:

When we submit an SQL Query to the Database then Database Engine will Perform the following Steps.

**Step 1- Query Tokenization:-** This Phase will take SQL Query as an Input,divided into no.of tokens andGenerate Stream of tokens as an output.

**Step 2- Query Proccessing:-** This phase will take Stream of tokens as an Input,constructs Query Tree with the Tokens,if Query Tree Success then no Syntax error is available in the provided SQL Query.If Query Tree is not Success then there are some syntax errors in the provided SQL Query.

**Step 3- Query Optimization:-** The main purpose of Query Optimization phase is to perform optimization on Query Tree in order to reduce execution time and to optimize memory utilization.

**Step 4- Query Execution:-** This phase will take optimized Query Tree as an input and execute the Query by using interpreters.

## JDBC (Java Database Connectivity):-

The process of interacting with the database from Java Applications is called as JDBC.

-->JDBC is an API,which will provide very good predefined library to connect with database from JAVAApplications in order to perform the basic database operations:

-->In case of JDBC Applications we will define the database logic and Java application and we will send a Java represented database logic to Database Engine.But database engine is unable to execute the Java represented database logic,it should required the database logic in Query Language Representations.

-->In the above context, to execute JDBC applications we should require a conversion mechanism to convert the database logic from Java representations to Query language representations and fromQuery language representations to Java representations.

-->In the above situation the required conversion mechanisms are available in the form of a software called as "Driver".

## Steps To design Jdbc Applications:-

1)Load and register the Driver.

2)Establish the connection between Java Application.

3)Prepare either Statement or preparedStatement or CallableStatement Objects.

4) Write and execute SQL Queries.

5) close the connection.

## 1. Load and register the driver:-

In general Driver is an interface provided by Sun Microsystems and whose implementation classes are provided by the Database Vendors as part of their Database Softwares.

-->To load and Register the Driver first we have to make available Driver implementation to

JDBC application.For this we have to set classpath environment variable to the location Where we have Driver implementation class.

-->If we want to use Type1 Driver provided by Sun MicroSystems in our JDBC applications then it is not required to set classpath environment variable because Type1 Driver was provided by Sun MicroSystems as part of Java Software in the form ofsun.jdbc.odbc.JdbcOdbcdriver

-->If we want to use Type1 Driver in our JDBC applications then before loading we have to

Configure the Microsoft product odbc Driver.

-->To configure Microsoft product odbc Driver we have to use the following path.

-->To load and register Driver in our Jdbc applications we have to use the following method from class 'Class'

Public static Class forName(String class_Name)

Eg:Class.forName("oracle.jdbc.OracleDriver");

-->When JVM encounter the above instruction JVM will pickup the parameter that is JDBCOdbcDriver Class name and JVM will search for its .class file in the current location,if it is not available then JVM will search for it in Java predefined library.

-->If JVM identify JDBCODBCDriver.class file in Java pre-defined library(rt.jar) then JVM will load JdbcOdbcDriver class byte code to the memory.

-->At the time of loading JdbcOdbcDriver class byte code to the memory JVM will execute a static block, As part of this JVM wiil execute a method call likeDriverManager.registerDriver(--); by the execution of registerDriver() method only JDBCOdbcDrvier will be available to our Jdbc applications.

-->In case of Type1 Driver if we use either Jdbc 4.0 version or Jdk 6.0 version then it is optional

To perform loading and register the driver step because JVM will perform Drvier registration

automatically at the time of establishing the connection between Java application and Database.

-->java.sql package includes the following pre-defined library to design Jdbc applications.

1. Driver (I)

2. DriverManager (C)

3. Connection (I)

4. Statement (I)

5. PreparedStatement (I)

6. ResultSet (I)

7. ResultSetMetaData (I)

8. DatabaseMetaData (I)

9. Savepoint(i)

## 2)Establish the Connection between Java application and Database:

-->To establish the connection between Java application and Database we have to use the following Method from DriverManager class.

Public static Connection getConnection(String url,String db_user_name,String db_password)

Ex:Connection con=DriverManager.getConnection("jdbc:odbc:dsnName","system","test");

-->When JVM encounter the above instruction JVM will access getConnection method,as part

of the getConnection method JVM will access connect() method to establish virtual socket connection Between Java application and database as per the url which we provided.

-->where getConnection() method will take three parameters 1.Driver URL 2.Database username 3.Database password -->In general from Driver to Driver Driver class name and Driver url will varied.

-->If we use Type1 Driver then we have to use the following Driver class name and URL

d-class : sun.jdbc.odbc.JdbcOdbcDriver url :jdbc:odbc:dsnName

-->In general all the Jdbc Drivers should have an url with the following format. main-protocol: sub-protocol

-->where main-protocol name should be Jdbc for each and every Driver but the sub protocol name should be varied from Driver to Driver.

## 3)Create either Statement or PreparedStatment or CallableStatement objects as per the requirement:

As part of the Jdbc applications after establish the connection between Java application and Database.

We have to prepare SQL Queries,we have to transfer SQL Queries to the databseEngine and we have to make Database Engine to execute SQL Queries.

-->To write and execute SQL Queries we have to use same predefined library from Statement preparedStatement and callableStatement.

-->To use the above required predefined library we have to prepare either Statement or

preparedStatement or CallableStatement objects.

**Q)What is the difference between Statement,PreparedStatement and Callable Statement Objects.**

**Ans:**

-->In Jdbc applications when we have a requirement to execute all the SQL Queries independently we have to use Statement.

-->In jdbc applications when we have a requirement to execute the same SQL Query in the next

Sequence where to improve the performance of JDBC application we will use prepared Statement.

-->In jdbc applications when we have a requirement to access stored procedures and functions available At Database from Java application we will use Callable Statement object.

-->To prepare Statement object we have to use the following method from Connection.

Public Statement createStatement()

Ex: Statement st=con.createStatement();

Where createStatement() method will return Statement object by creating Statement interfaces Anonymous inner class object.

## 4)Write and execute SQL Queries:

1.executeQuery()

2.executeUpdate()

3.execute()

**Q)What are the differences between executeQuery(),executeUpdate() and execute() method?**

**Ans:**Where executeQuery() method can be used to execute “selection group SQL Queries” in order to fetch(retrieve) data from Database.

-->when JVM encounter executeQuery() method with selection group SQL query then JVM will pickup Selection group SQL Query,send to JdbcOdbcDriver,it will send to connection.Now connection will carrythat SQL Query to the database engine through Odbc Driver.

-->At database database engine will execute the selection group SQL Query by performing Query Tokenization,Query parsing,Query optimization and Query Execution.

-->By the execution of selection group SQL Query database engine will fetch the data from database and return to Java Application.

-->As Java technology is pure object oriented,Java application will store the fetched data in the form of an object at heap memory called as “ResultSet”.

-->As per the predefined implementation of executeQuery method JVM will return the generated ResultSet object reference as return value from executeQuery() method.

Public ResultSet executeQuery(String sql_Query) throws SQLException

Ex: ResultSet rs=st.executeQuery(“select * from emp1”);

-->where executeUpdate() method can be used to execute updation group SQLQueries in order to perform the database operations like create,insert,update,delete,Drop…. -->when JVM encounter updation group SQL Query with execteUpdate() method the JVM will pickup

That Sql Query and send to Database through Database connection.At Database side Database engine will execute it,perform updation from Database,identify rowCount value (number of records got updated) and return to Java application.

-->As per the predefined implementation of executeUpdate() method JVM will return row count value From executeUpdate() method.

Public int executeUpdate(String sql_Query) throws Exception

Ex: int rowCount=st.executeUpdate("update emp1 set esal=esal+500 where esal<1000");

-->In Jdbc applications execute() method can be used to execute both selection group and updation Group SQL Queries.

-->when JVM encounter selection group SQL Query with execute() method then JVM will send selection Group SQL Query to database engine,where database engine will execute it and send back the fetched Data to Java Application.

-->In Java application ResultSet object will be created with the fetched data but as per the predefined implementation of execute() method JVM will return "true" as a Boolean value.

-->when JVM encounter updation group SQL Query as parameter to execute() method then JVM Will send it to the database engine,where database engine will perform updations on database and return row Count value to Java application.But as per the predefined implementation of execute() method JVM will return "false" as Boolean value from execute() method.

public Boolean execute(String sql_Query) throws SQLException.

Ex: boolean b1=st.execute ("select * from emp1");

boolean b2=st.execute("update emp1 set esal=esal+500 where esal<10000");

## 5)Close the connection:

In Jdbc applications after the database logic it is convention to close Database connection for this we have to used the following method.

Public void close() throws SQLException

Ex:con.close();

**JdbcApp1: The following example demonstrate how to create a table on Database through a JDBC applicationby taking table name as Dynamic input.**

```
//import section import java.sql.*; import java.io.*; class JdbcApp1{ public static void
main(String args[]) throws Exception{

//load a register driver

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

//establish connection between Java application and database  Connection
con=DriverManager.getConnection("jdbc:odbc:nag","system","durga");

//prepare Statement Statement st=con.createStatement();

//create BufferedReader BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));

//take table name as dynamic input

System.out.println("Enter table name");

String tname=br.readLine();

//prepare SQLQuery

String sql="create table " + tname + "(eno number,ename varchar2(10),esal number)";

//execute SQL Query

st.executeUpdate(sql);
System.out.println("table created successfully");

//close the connection

con.close(); }}
```

**JdbcApp2: The following example demonstrates how to insert no.of records on database table by taking records data as dynamic input.**

```
import java.io.*;
import java.sql.*;
public class JdbcApp2 {
public static void main(String[] args)throws Exception {
Class.forName("oracle.jdbc.OracleDriver");
Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"system","test");
Statement st=con.createStatement();
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
while(true)   {
System.out.print("Employee Number   :");
int eno=Integer.parseInt(br.readLine());
System.out.print("Employee Name     :");
String ename=br.readLine();
System.out.print("Employee Salary   :");
float esal=Float.parseFloat(br.readLine());
System.out.print("Employee Address  :");
String eaddr=br.readLine();
st.executeUpdate("insert into emp1  values("+eno+",'"+ename+"',"+esal+",'"+eaddr+"')");
System.out.println("Employee Inserted Successfully");
System.out.print("Onemore Employee[Yes/No]?  :");
String option=br.readLine();    .
if(option.equals("No"))
{
break;
}
}
con.close();
}
}
```

**JdbcApp3: The following example Demonstrate how to perform updations on Database table through Jdbc Application.**

```
import java.io.*;
import java.sql.*;
```

```
public class JdbcApp3 {
public static void main(String[] args)throws Exception {
Class.forName("oracle.jdbc.OracleDriver");
Connection   con=DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:xe",
"system","test");
Statement st=con.createStatement();
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.print("Bonus Amount  :");
int bonus_Amt=Integer.parseInt(br.readLine());
System.out.print("Salary Range  :");
float sal_Range=Float.parseFloat(br.readLine());
int rowCount=st.executeUpdate ("update emp1 set esal=esal+"+bonus_Amt+" where
esal<"+sal_Range);
System.out.println("Employees Updated  :"+rowCount);   con.close();
}
}
```

**JdbcApp4: The following example demonstrates how to delete no.of records from database table through a Jdbc application**

```
import java.io.*;
import java.sql.*;
public class JdbcApp4 {
public static void main(String[] args)throws Exception {
DriverManager.registerDriver(new oracle.jdbc.OracleDriver());
Connection   con=DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:xe",
"system","test");
Statement st=con.createStatement();
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.print("Salary Range   :");
float sal_Range=Float.parseFloat(br.readLine());
int rowCount=st.executeUpdate("delete from emp1 where esal<"+sal_Range);
System.out.println("Records Deleted   :"+rowCount);
con.close();
}
}
```

# Concept of JSP (Java Server Pages)

**Introduction:-** The main purpose of the web applications in Enterprise applications area is to generate dynamic response from server machine.

To design web applications at server side we may use Web Technologies like CGI, Servlets, JSP and so on.

CGI is basically a Process based technology because it was designed on the basis of C technology.

If we deploy any CGI application at server then for every request CGI container will generate a separate process.

In the above context, if we send multiple number of requests to the same CGI application then CGI container has to generate multiple number of processes at server machine.

To handle multiple number of processes at a time server machine has to consume more number of system resources, as a result the performance of the server side application will be reduced.

To overcome the above problem we have to use Thread based technology at server side like servlets.

In web application development, servlets are very good at the time of pick up the request and process the request but servlets are not good at the time of generating dynamic response to client.

Servlet is a Thread based technology, if we deploy it at server then container will create a separate thread instead of the process for every request from the client.

Due to this Thread based technology at server side server side application performance will be increased.

In case of the servlet, we are unable to separate both presentation logic and business logic.

If we perform any modifications on servlets then we must perform recompilation and reloading.

If we want to design web applications by using servlets then we must require very good knowledge on Java technology

JSP is a server side technology provided by Sun Microsystems to design web applications in order to generate dynamic response.

The main intention to introduce Jsp technology is to reduce java code as much as possible in web applications.

Jsp technology is a server side technology, it was designed on the basis of Servlet API and Java API.

In web application development, we will utilize Jsp technology to prepare view part or presentation part.

Jsp technology is very good at the time of generating dynamic response to client with very good look and feel.

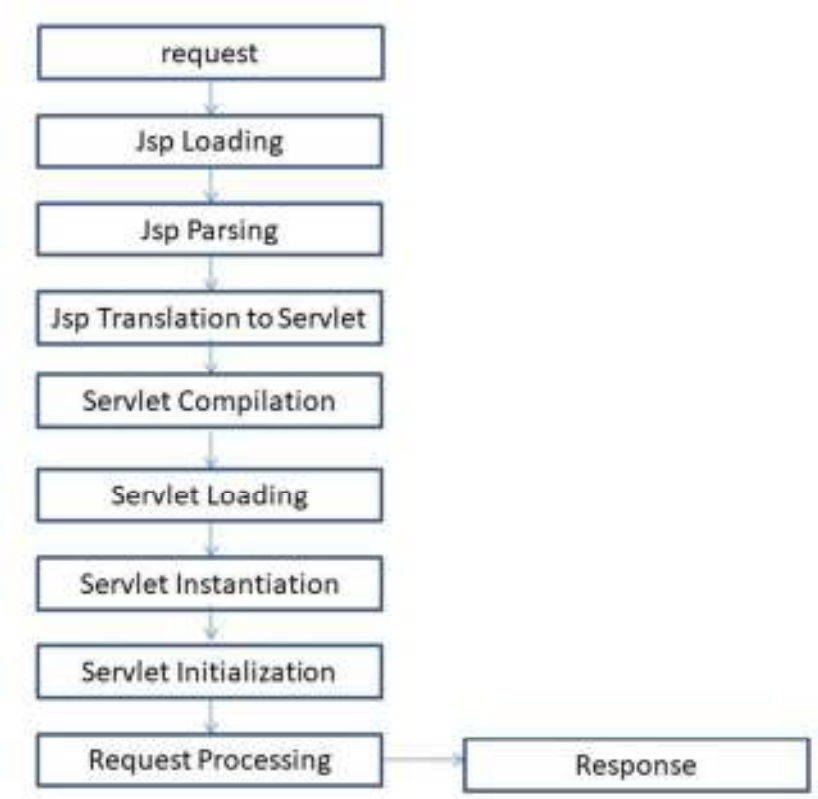
If we want to design any web application with Jsp technology then it is not required to have java knowledge.

In case of Jsp technology, we are able to separate presentation logic and business logic because to prepare presentation logic we will use html tags and to prepare business logic we will use Jsp tags separately.

If we perform any modifications on Jsp pages then it is not required to perform recompilation and reloading because Jsp pages are auto-compiled and auto-loaded.

## JSP Life Cycle:-




When we send request from client to server for a particular Jsp page then container will pick up the request, identify the requested Jsp pages and perform the following life cycle actions.

**JSP Loading:-** Here container will load Jsp file to the memory from web application directory structure.

**JSP Parsing:-** Here container will check whether all the tags available in Jsp page are in well-formed format or not.

**JSP Translation To Servlet:-** After the Jsp parsing container will translate the loaded Jsp page into a particular servlet. While executing a Jsp page Tomcat container will provide the translated servlet in the following location at Tomcat Server.

**Servlet Compilation:-** After getting the translated servlet container will compile servlet java file and generates the respective .class file.

**Servlet Loading:-** Here container will load the translated servlet class byte code to the memory.

**Servlet Instantiation:-** Here container will create object for the loaded servlet.

**Servlet Initialization:-** Here container will access _JspInit() method to initialize the servlet.

**Creating request and response objects:-** After the servlet initialization container will create a thread to access _JspService(_,_) method, for this container has to create HttpServletRequest and HttpServletResponse.

**Generating Dynamic response:-** After getting request and response objects container will access _JspService(_,_) method, by executing its content container will generate some response on response object.

**Dispatching Dynamic response to Client:-** When container generated thread reached to the ending point of _JspService(_,_) method then that thread will be in Dead state, with this container will dispatch dynamic response to client through the Response Format prepared by the protocol.

**Destroying request and response objects:-** When the dynamic response reached to client protocol will terminate its virtual socket connection, with this container will destroy request and response objects.

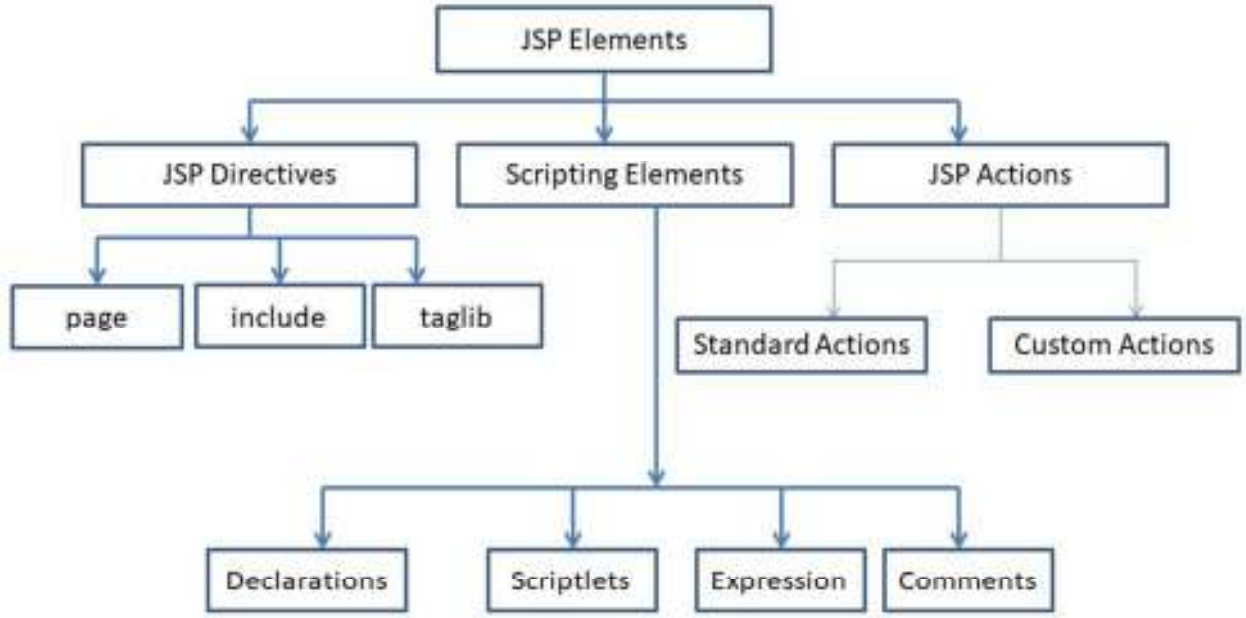
**Servlet Deinstantiation:-** After destroying request and response objects container will be in waiting state depends on the container, then container identifies no further request for the same resource then container will destroy servlet object, for this container will execute _JspDestroy() method.

**Servlet Unloading and Jsp Unloading:-** After the servlet deinstantiation container will eliminate the translated servlet byte code and Jsp code from memory.

## Lecture -15

# JSP Elements




**Q: What are the differences between Jsp Directives and Scripting Elements?**

**Ans:**

1. In web applications, Jsp Directives can be used to define present Jsp page characteristics, to include the target resource content into the present Jsp page and to make availableuser defined tag library into the present Jsp page.

In web applications, Jsp Scripting Elements can be used to provide code in Jsp pages.

2. All the Jsp Directives will be resolved at the time of translating Jsp page to servlet.

All the Jsp Scripting Elements will be resolved at the time of request processing.

3. Majority of the Jsp Directives will not give direct effect to response generation, but majority of Scripting Elements will give direct effect to response generation.

**Q: To design Jsp pages we have already Jsp Scripting Elements then what is requirement to go for Jsp Actions?**

**Ans:** In Jsp applications, Scripting Elements can be used to allow java code inside Jsp pages but the main theme of Jsp technology is not to allow java code inside the Jsp pages.

In the above context, to preserve the theme ofJsp technology we have to eliminate scripting elements from Jsp pages, for this we have to provide an alternative i.e. Jsp Actions provided by Jsp technology.

In case of Jsp Actions, we will define scripting tag in place of java code, in Jsp pages and we will provide the respective java code inside the classes folder.

In this context, when Jsp container encounter the scripting tag then container will execute the respective java code and perform a particular action called as Jsp Action.

## JSP Directives

To provide Jsp Directives in Jsp pages we have to use the following syntaxes.

### 1. Jsp-based Syntaxes:-

<%@Directive_name [attribute-list]%>

Ex: %@page import=”java.io.*”%

### 2. XML based Syntaxes:-

<jsp:directive.directiveName[attr ibute-list]%/>

Ex: <jsp:directive.page import=”java.io.*”/>

There are 3 types of Directives in Jsp technology.

1. Page Directive

2. Include Directive

3. Taglib Directive

**1. Page Directive:-** In Jsp technology, Page Directive can be used to define the present Jsp page characteristics like to define import statements, specify particular super class to the translated servlet, to specify metadata about present Jsp pages and so on.

**Syntax 1:**<%@page [attribute-list]%>

**Syntax 2:**<jsp:directive.page [attribute-list]/>

Where attribute-list in Jsp page directive may include the following list.

1. language

2. contentType

3. import

4. extends

5. info

6. buffer

7. autoFlush

8. errorPage

9. isErrorPage

10. session

11. isThreadSafe

12. isELIgnored

**1. Language:-** This attribute can be used to specify a particular scripting language to use scripting elements.

The default value of this attribute is java.

Ex: <%@page language="java"%>

**2.contentType:-** This attribute will take a particular MIME type in order to give an intimation to the client about to specify the type of response which Jsp page has generated.

Ex: <%@page contentType="text/html"%>

**3.import:-** This attribute can be used to import a particular package/packages into the present Jsp pages.

Ex:<%@page import="java.io.*"%>

If we want to import multiple number of packages into the present Jsp pages then we have to use either of the following 2 approaches.

**Approach 1:**

Specify multiple number of packages with comma(,) as separator to a single import attribute.

Ex: <%@page import="java.io.*,java.util.*,java.sql.*"%>

**Approach 2:**

Provide multiple number of import attributes for the list of packages.

Ex: %@page import="java.io.*" import="java.util.*" import="java.sql.*"%

**4.extends:-** This attribute will take a particular class name, it will be available to the translated servlet as super class.

Ex: <%@page extends="com.dss.MyClass"%>

Where MyClass should be an implementation class to HttpJspPage interface and should be a subclass to HttpServlet.

The default value of this attribute is HttpJspBase class.

**5.info:-** This attribute can be used to specify some metadata about the present Jsp page.

Ex: <%@page info="First Jsp Application"%>

If we want to get the specified metadata programmatically then we have to use the following method from Servlet interface.

public String getServletInfo()

The default value of this attribute is Jasper JSP2.2 Engine.

**6.Buffer:-** This attribute can be used to specify the particular size to the buffer available in

JspWriter object.

Note: Jsp technology is having its own writer object to track the generated dynamic response, JspWriter will provide very good performance when compared with PrintWriter in servlets.

Ex: <%@page buffer="52kb"%>

The default value of this attribute is 8kb.

**7.autoFlush:-** It is a boolean attribute, it can be used to give an intimation to the container about to flush or not to flush dynamic response to client automatically when JspWriter buffer filled with the response completely.

If autoFlush attribute value is true then container will flush the complete response to the client from the buffer when it reaches its maximum capacity.

If autoFlush attribute value is false then container will raise an exception when the buffer is filled with the response.

**8.errorPage:-** This attribute can be used to specify an error page to execute when we have an exception in the present Jsp page.

Ex: %@page errorPage=”error.jsp”%

**9.isErrorPage:-** It is a boolean attribute, it can be used to give an intimation to the container about to allow or not to allow exception implicit object into the present Jsp page.

If we provide value as true to this attribute then container will allow exception implicit object into the present Jsp page.

If we provide value as false to this attribute then container will not allow exception implicit object into the present Jsp page.

The default value of this attribute is false.

Ex: <%@page isErrorPage=”true”%>

**first.jsp:**

```
<%@page errorPage="error.jsp"%>
<%
java.util.Date d=null;
out.println(d.toString());
%>
```

**error.jsp:**

```
<%@page isErrorPage="true"%>
<html>
<body bgcolor="lightgreen">
<center><b><font size="" color=""><br><br>
<%=expression%>
</font></b></center></body>
</html>
```

**10.session:-** It is a boolean attribute, it is give an intimation to the container about to allow or not to allow session implicit object into the present Jsp page.The default value of this attribute is true.

Ex: <%@page session=”true”%>

**11. isThreadSafe:-** It is a boolean attribute, it can be used to give an intimation to the container about to allow or not to allow multiple number of requests at a time into the present Jsp page.

If we provide true as value to this attribute then container will allow multiple number of requests at a time.

If we provide false as value to this attribute then automatically container will allow only one request at a time and it will implement SingleThreadModel interface in the translated servlet.

The default value of this attribute is true.

Ex: <%@page isThreadSafe=”true”%>

**12.isELIgnored:-** It is a boolean attribute, it can be used to give an intimation to the container about to allow or not to allow Expression Language syntaxes in the present Jsp page.

Note: Expression Language is a Scripting language, it can be used to eliminate java code completely from the Jsp pages.

If isELIgnored attribute value is true then container will eliminate Expression Language syntaxes from the present Jsp page.

If we provide false as value to this attribute then container will allow Expression Language syntaxes into the present Jsp pages.

The default value of this attribute is false.

Ex: <%@page isELIgnored=”true”%>

2.Include Directive:- Include Directive can be used to include the content of the target resource into the present Jsp page.

Syntax:<%@include file=”--”%>

Where file attribute can be used to specify the name and location of the target resource.

**logo.jsp:**

```
 <html>
<body>
<center>
<table width="100%" height="20%" bgcolor="red">
<tr><td colspan="2"><center><b><font size="7" color="white"> Softpro India
</font></b></center></td></tr>
</table>
</center>
</body>
</html>
```

**footer.jsp:**

```
<html>
<body>
<center>
<table width="100%" height="15%" bgcolor="blue">
<tr><td colspan="2"><center><b><font size="6" color="white"> copyrights2018-
2020@softproindia </font></b></center></td></tr> </table>
</center>
</body>
</html>
```

**body.jsp:**

```
<html>
<body bgcolor="lightyellow">
<center><b><font size="7"> <p><br>      Softpro India Computer Technologies Pvt. Ltd.
<br><br></p> </font></b>
</center>
</body>
</html>
```

### mainpage.jsp:

```
<%@include file="logo.jsp"%>
<%@include file="body.jsp"%>
<%@include file="footer.jsp"%>
```

### 3.Taglib Directive:-

The main purpose of Taglib Directive is to make available user defined tag library into the present Jsp pages.

Syntax:<%@taglib uri="_ _" prefix="_ _"%>

Where uri attribute can be used to specify the name and location of user defined tag library.

Where prefix attribute can be used to define prefix names for the custom tags.

Ex: %@taglib uri="/WEB-INF/db.tld" prefix="connect"%

## JSP Scripting Elements

The main purpose of Jsp Scripting Elements is to allow java code directly into the present Jsp pages.

There are 3 types of Scripting Elements.

1. Declarations

2. Scriptlets

3. Expressions

### 1. Declarations:-

It can be used to provide all the java declarations like variable declarations, method definitions, classes declaration and so on.

**Syntax:**<%! --------

--------

--------

%>

If we provide any java declarations by using declaration scripting element then that all java declarations will be available to the translated servlet as class level declarations.

**2.Scriptlets:-** This Scripting Element can be used to provide a block of java code.

**Syntax:**<%

--------

--------

--------

%>

If we provide any block of java code by using scriptlets then that code will be available to translated servlet inside _jspService(_,_) method.

**3.Expressions:-** This is scripting element can be used to evaluate the single java expression and display that expression resultant value onto the client browser.

**Syntax:**<%=Java Expression%>

If we provide any java expression in expression scripting element then that expression will be available to translated servlet inside _JspService(_,_) method as a parameter to out.write() method.

Ex: out.write(Java Expression);

## JSP Implicit Objects

In J2SE applications, for every java developer it is common requirement to display data on command prompt.

To perform this operation every time we have to prepare PrintStream object with command prompt location as target location

PrintStream ps=new PrintStream("C:\program Files\.....\cmd.exe") ;

ps.println("Hello");

In java applications, PrintStream object is frequent requirement so that java technology has provided that PrintStream object as predefined object in the form of out variable in System class.

public static final PrintStream out;

Similarly in web applications, all the web developers may require some objects like request, response, config, context, session and so on are frequent requirement, to get these objects we have to write some piece of java code.

Once the above specified objects are as frequent requirement in web applications, Jsp technology has provided them as predefined support in the form of Jsp Implicit Objects in order to reduce burden on the developers.'

Jsp technology has provided the following list of implicit objects with their respective types.

1. out---→ javax.servlet.jsp.JspWriter

2. request---→ javax.servlet.HttpServletRequest

3. response ---→ javax.servlet.HttpServletResponse

4. config ---→ javax.servlet.ServletConfig

5. application ---→ javax.servlet.ServletContext

6. session ---→ javax.servlet.http.HttpSession

7. exception ---→ java.lang.Throwable

8. page ---→ java.lang.Object

9. pageContext ---→ javax.servlet.jsp.PageContext

**Q: What is the difference between PrintWriter and JspWriter?**

**Ans:** PrintWriter is a writer in servlet applications, it can be used to carry the response. PrintWriter is not BufferedWriter so that its performance is very less in servlets applications.

JspWriter is a writer in Jsp technology to carry the response. JspWriter is a BufferedWriter so that its performance will be more when compared with PrintWriter.

**Q: What is PageContext? and What is the purpose of PageContext in Jsp Applications?**

**Ans:** PageContext is an implicit object in Jsp technology, it can be used to get all the Jsp implicit objects even in non-jsp environment.

To get all the Jsp implicit objects from PageContext we have to use the following method.

public Xxx getXxx()

Where Xxx may be out, request, response and so on.

Ex: JspWriterout=pageContext.getOut();

HttpServletRequest request=pageContext.getRequest();

ServletConfig config=pageConetxt.getServletConfig();

**Note:** While preparing Tag Handler classes in custom tags container will provide pageContext object as part of its life cycle, from this we are able to get all the implicit objects.

In Jsp applications, by using pageContext object we are able to perform some operations with the attributes in Jsp scopes page, request, session and application like adding an attribute, removing an attribute, getting an attribute and finding an attribute.

To set an attribute onto a particular scope pageContext has provided the following method.

public void setAttribute(String name, Object value, int scope)

Where scopes may be

public static final int PAGE_SCOPE=1;

public static final int REQUEST_SCOPE=2;

public static final int SESSION_SCOPE=3;

public static final int APPLICATION_SCOPE=4;

**Ex:** <% pageContext.setAttribute(“a”, “aaa”, pageConext.REQUEST_SCOPE); %>

To get an attribute from page scope we have to use the following method.

public Object getAttribute(String name)

**Ex:** String a=(String)pageContext.getAttribute(“a”);

If we want to get an attribute value from the specified scope we have to use the following method.

public Object getAttribute(String name, int scope)

**Ex:** String uname=(String)pageContext.getAttribute("uname", pageContext.SESSION_SCOPE);

To remove an attribute from a particular we have to use the following method.

public void removeAttribute(String name, int scope)

**Ex:** pageContext.removeAttribute("a", pageContext.SESSION_SCOPE);

To find an attribute value from page scope, request scope, session scope and application scope we have to use the following method.

Public Object findAttribute(String name)

Ex: String name=pageContext.findAttribute("uname");

## JSP Scopes

In J2SE applications, to define data availability i.e. scope for we have to use access specifierspublic, protected, default and private.

Similarly to make available data to number of resources Jsp technology has provided the following 4 types of scopes with the access modifiers.

1. **Page Scope:-** If we declare the data in page scope by using pageContext object then that data should have the scope upto the present Jsp page.
2. **Request Scope:-** If we declare the data in request object then that data should have the scope up to the number of resources which are visited by the present request object.
3. **Session Scope:-** If we declare the data in HttpSession object then that data should have the scope up to the number of resources which are visited by the present client.
4. **Application Scope:-** If we declare the data in ServletContext object then that data should have the scope up to the number of resources which are available in the present web application.

## employeedetails.html:

```
<html>
<body bgcolor="lightblue">
<br><br><br><br>
<center>
<h1>Employee Details Form</h1>
</center>
<pre>
<h2>
<form method="get" action="display.jsp">
Employee Id :<input type="text" name="eid"/>
Employee Name : <input type="text" name="ename"/>
Employee Salary : <input type="text" name="esal"/>
<input type="submit" value="Display"/>
</h2>
</pre>
</body>
</html>
```

## Display.jsp:

```
<%!
int eid;
String ename;
float esal;
%>
<%
try {
int eid=Integer.parseInt(request.getParameter("eid"));
String ename=request.getParameter("ename");
float esal=Float.parseFloat(request.getParameter("esal"));
}
catch(Exception e){
e.printStackTrace();
}
%>
<html>
<body>
<center>
<h1>Employee Details</h1>
</center>
```

```
<center>
Employee Id : <%=eid %><br><br>
Employee Name : <%=ename %><br><br>
Employee Salary : <%=esal %><br><br>
</center>
</body>
</html>
```

## JSP Actions

In Jsp technology, by using scripting elements we are able to provide java code inside the Jsp pages, but the main theme of Jsp technology is not to allow java code inside Jsp pages.

To eliminate java code from Jsp pages we have to eliminate scripting elements, to eliminate scripting elements from Jsp pages we have to provide an alternative i.e. Jsp Actions.

In case of Jsp Actions, we will define a scripting tag in Jsp page and we will provide a block of java code w.r.t. scripting tag.

When container encounters the scripting tag then container will execute respective java code, by this an action will be performed called as Jsp Action.

In Jsp technology, there are 2 types of actions.

1. Standard Actions

2. Custom Actions

# Spring Framework Contents

- ➔ C, C++, Java are programming language.
- ➔ JDBC, Servlet, EJB, JSP, JMS, Java Mail, JTA, JPA and etc are Java Technologies.
- ➔ Struts, Spring, Hibernate and etc are Java Frameworks.

## Programming Languages:-

It is directly installed software having features to develop software applications.

It defines syntaxes (Rules) and symantics (Structures) of programming.

These are base to create software technologies, frameworks, tools, Database software, operating system and etc..

These are raw materials of software development.

E.g. C, C++, C#, Java.

## Software Technology:-

It is software specification having set of rules and guidelines to develop software by using programming language support.

Software Technology is not installable but software created based on software technology is installable.

In java environment, technology rules are nothing but interfaces and guidelines are nothing but classes. Abstract class represent both rules and guidelines.

e.g. JDBC, JSP, Servlet and etc..

JDBC technology gives rules and guidelines to develop driver using java language. Working with JDBC Driver is nothing but working with JDBC technologies.

There are 2 types of software technologies.

1. Open Technologies.
2. Proprietary Technologies.

## Open Technologies:-

Here the rules and guidelines of technology are open to all software vendor companies to develop software.

E.g. All Java Technologies (JDBC, Servlet, JSP etc.)

### Proprietary Technologies:-

Here the Vendor Company who has given technology is only allowed to develop software based on technologies.

E.g. All Microsoft Technologies (Like ASP.Net).

### Frameworks:-

Framework is installable software that uses existing technologies internally to simplify the application development having ability to generate common logic of application internally. Framework is installable software that provides abstraction layer on existing technologies to simplify the application development process.

E.g. Struts, spring, Hibernate …

While working with technologies the programmer has develop both common logic and specific logics of application.

While working with frameworks the programmer has to develop only application specific logics because the common logics will be generated by the framework dynamically.

Every framework uses technology internally but it never makes programmer to know about it. This is nothing but providing abstraction layer (hiding implementations/internals).

### Plain JDBC App:-

- ➔ Register JDBC Driver (load Driver class)
- ➔ Establish the Connection
- ➔ Create Statement Object
- ➔ Send and execute SQL query in Database software.
- ➔ Gather results and process results.
- ➔ Close JDBC objects.

### Spring JDBC App:-

- ➔ Get JdbcTemplate object.
- ➔ Send and execute SQLQuery in Database software.
- ➔ Gather results and process results.

While working with technology we need to write some common logics in every application of that technology. This is called ***boiler plate code*** problem.

While working with frameworks we just need to develop application specific logics that means it solves ***boiler plate code*** problem.

**Based on kinds of application and logics we develop, there are 3 types of Java Frameworks.**

## Web Application Frameworks:-

Provides Abstraction layer on servlets, jsp technology.

Simplifies MVC Architecture based web application development.

JSF -> From Sun MS (Oracle)

Struts-> From Apache

Webwork-> From open symphony.

ADF-> From Oracle

Spring MVC-> From interface

## ORM Frameworks:-

Provides Abstraction layer on JDBC technology.

Simplifies Object based O-R mapping persistent logic development.

Allows to portability/Database independent persistent logic.

Hibernate-> from Softtraa (RedHat)

TOPLink-> Oracle Corporation

OJB-> Apache

JDO-> from Adobe

## Application Frameworks:-

Provides abstraction layer on multiple technologies like JDBC, Servlets, JSP, JNDI, JMS, JTA, JAAS, EJB, RMI, and etc.

Allows to develop all kinds of logic like business logic, persistence logic, presentation logic, integration logic and etc.

Allows to develop all kinds of applications like standalone applications, web applications, distributed applications and etc.

E.g. Spring -> from interfaces

**Based on mode of development they allow, there are 2 kinds of framework.**

## Invasive Framework:

Here the classes of application should extend from framework api class or should implement framework api interface.

This process make api classes as tightly bounded classes of frameworks.

We can't change framework of application/ project by changing the libraries.

## Non-Invasive Framework:

Here the class of application need not to extend from framework class or need not to implement framework api interfaces. So our application classes are not tightly bound with framework. We can change framework of application/ project by changing the libraries (jar files).

E.g. Struts 2.x, Spring, Hibernate and etc.

## Spring (2003):-

Initially for business logic development with ordinary classes allowing to apply middleware services.

Now Spring can be used to develop all kinds of logics and applications in light weight environment.

## Before Framework:-

Programmer directly uses technologies to develop both common and specific logics of application.

## With Framework:-

Programmers uses frameworks to develop both common and specific logics of application.

**Question:-** Is the Spring replacement for Struts?

**Answer:-** Using Struts we can develop only web application. There is no provision in Struts to develop and use middleware services / aspects (security, logging and etc..)

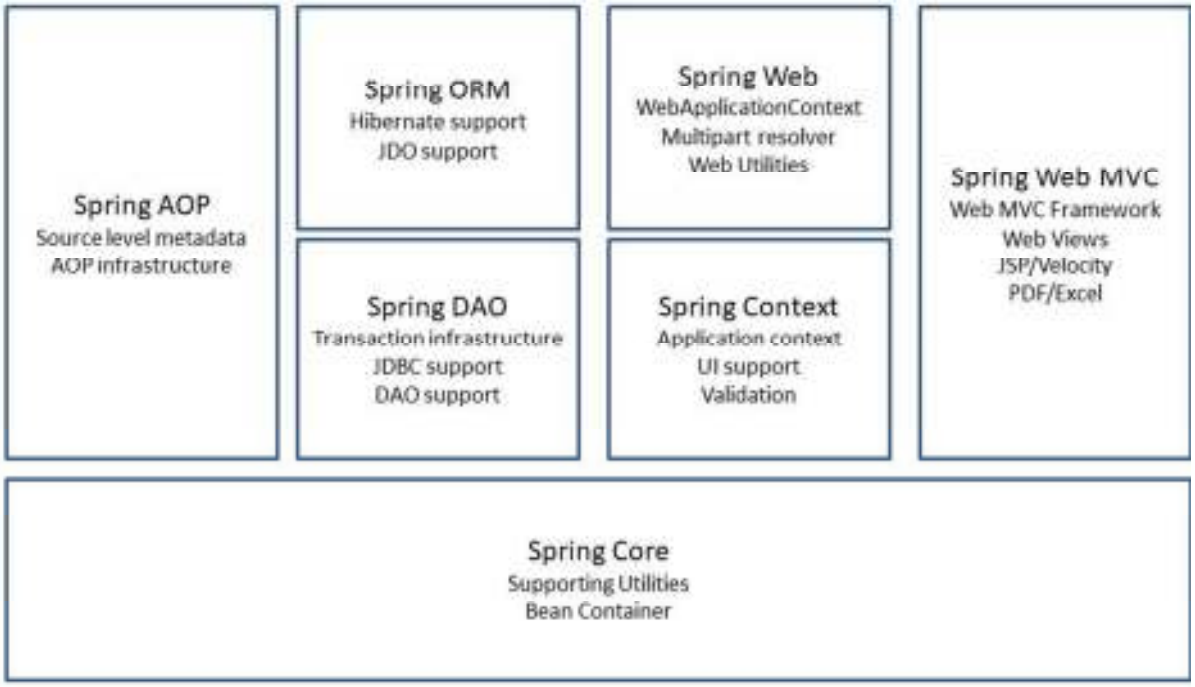
Using Spring we can develop all kinds of applications and all kinds of logics including web applications. It also provide environment to develop aspects/middleware services. Spring is powerfull than Struts.

**Question:-** Is the Spring replacement for JEE / JSE technologies?

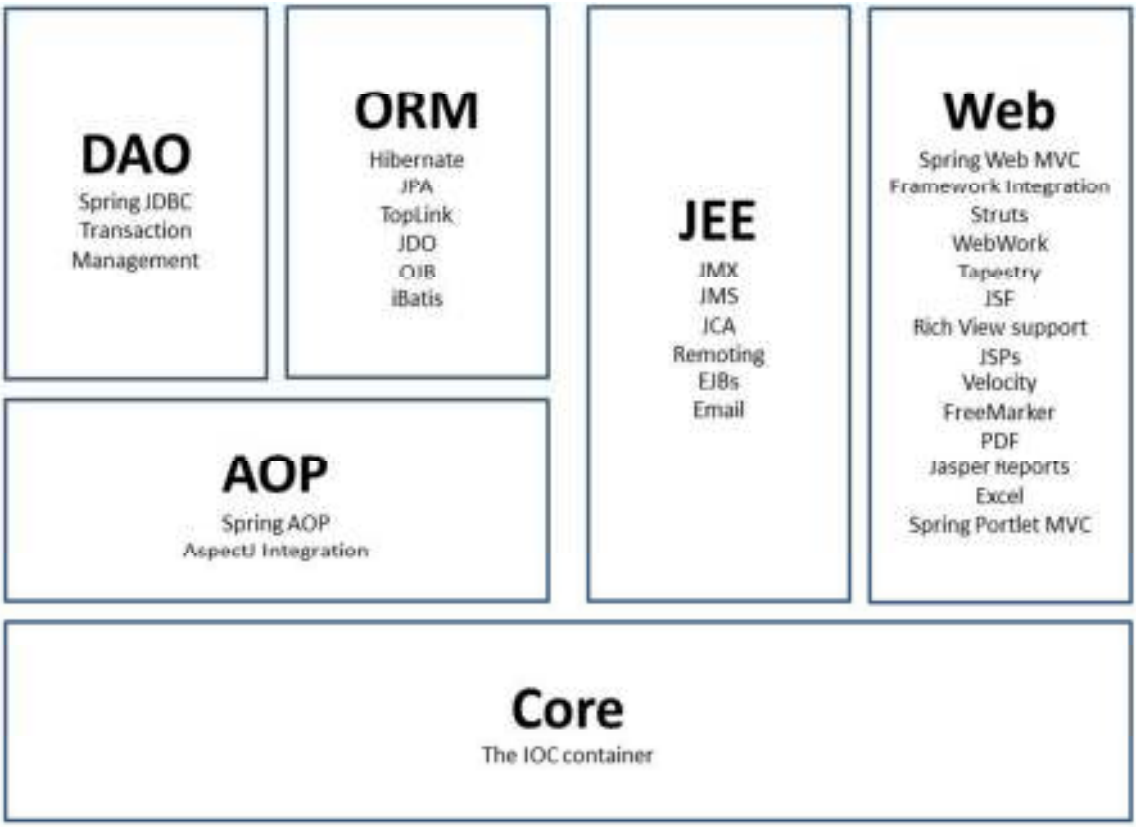
**Answer:-** No. Spring complement JEE / JSE technologies by using them internally in application development process.

**Spring 1.X Modules (7 Modules)**




**Spring 2.X Overview Diagram:**



**Spring 2.x web module :** Spring 1.x web + spring 1.x web mvc module.

### Spring Core:-

This module is base for other modules , provides containers that required to manage various spring components/resources.

Useful to develop standalone applications, Designed supporting dependency injection (injecting values to objects dynamically).

### Spring DAO/JDBC:-

Provides abstraction layer on plain JDBC to simplify the JDBC style persistence logic and ***Exception Re-Throwing*** concept is used to designed Exception herirachy.

### Spring ORM:-

Provides abstraction layer on multiple ORM framework (like Hibernate) and simplifies Object Based O-R mapping persistence logic.

### Spring Web:-

Provides plug-in to make spring applications communicatable from other web framework applications like Struts, JSF and etc.. applications.

### Spring Web MVC:-

Spring own web framework to develop MVC architecture based web applications with plan classes by using orbiter methods of the classes.

### Spring Context:-

Provides abstraction layer on multiple JSE, JEE module technologies to simplify the development of common logics.

### Spring AOP:-

Provides a new methodology of programming to develop aspect/ middleware services logics and to apply them on spring applications.

**Before AOP:**

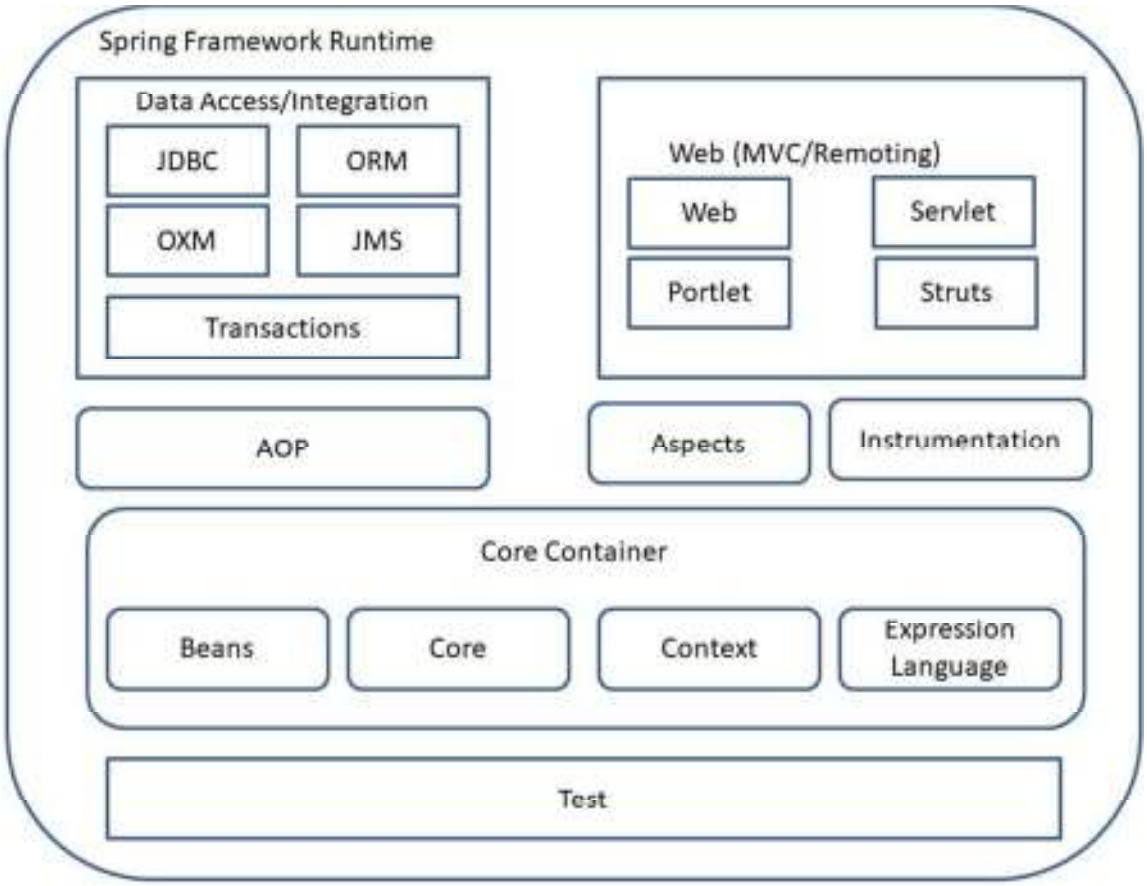
```
class Bank{
public void withdraw(_,_){
Security logic
Logging logic
Transaction logic
withdraw logic(bal=bal-amt)
}
public void deposite(){
Security logic
Logging logic
Transaction logic
 deposite logic(bal=bal+amt)
}}
```

**With AOP:**

```
class Bank{
public void withdraw(_,_){
withdraw logic(bal=bal-amt)
}
public void deposite(){
deposite logic(bal=bal+amt)
}}
```

## Spring 3.x Overview Diagram:




Spring 3.X is having 20 modules that are grouped into core container, AOP, instrumentation, Test, Data Access / Integration, MVC/ remoting groups.

Test module allows to perform unit testing (programmers testing on his own piece of code). Junit does not allow to mock objects (dummy objects), where spring 'Test' module allows us to work with mock object.

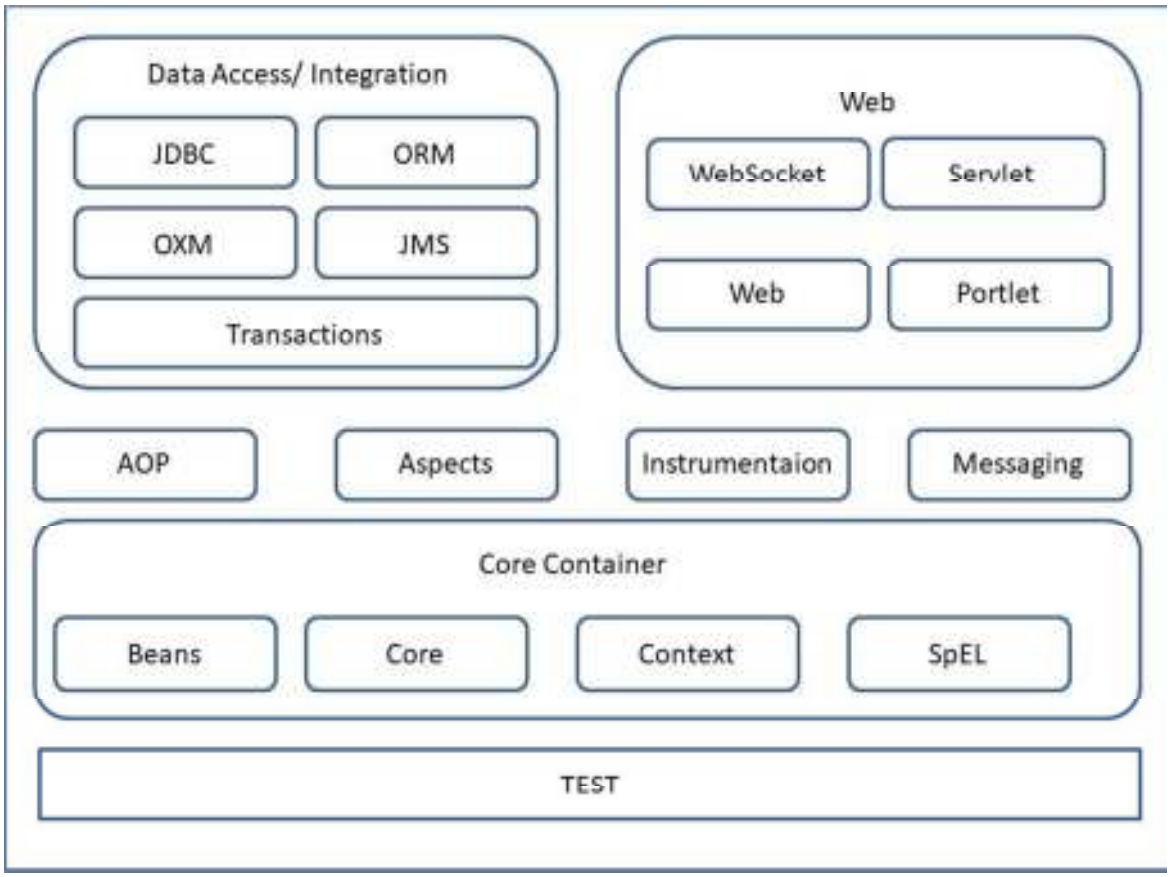
OXM (Object xml mapping) is given to convert java object to xml tags and vice-versa (alternate to JAXB).

Spring 2.x JEE services they placed in DataAccess/integration, MVC/remoting groups.

Portlet supports allows us to develop "portal" applications multiple windows (web pages) in a web page.

## Spring 4.x override diagram:




Spring 3.x is compatible with jdk 1.5.

Spring 4.x is compatible with jdk 1.6 and support all jdk 1.8 features. In this version all deprecated classes , methods are removed.

In Spring 4.x messaging is a sub framework in spring that given alternate to 'jms' to get messages based asynchronous communication between components.

Asynchronous communication allows the client to given next request or to client side operations without given request related response from server.

Web socket is a web level protocol that is designed on the top of http protocol for full 2-way communication.

Industry uses **"MVC" as defacto architecture** to develop java based medium/large scale web applications.

M-> Model -> Data + Business Logic + persistence logic (like account officer)

V-> View -> presentation logic (like Beautician)

C-> Controller -> Integration logic (like Traffic police/supervisor)

Note:- Integration logic controls and monitors all the activities of application execution. It is responsible to get communication between view layer and model layer components.

## DAO: Data Access Object:

The java class that separates persistence logic from other logics and makes the persistence logic as reusable logic and flexible logic to modify is called DAO.

JDBC is light weight but does not allow to develop object based persistence logic. EJB entity Beans heavy weight but allows to develop object based persistence logic. To overcome both problems we can use "Hibernate".

## Spring:-

|------→ **Type :** Java application framework

|-----→ **Version:** 4.x (Compactable with jdk 1.6+)

3.x (Compactable with jdk 1.5+)

|-----→ **Vendor :** interface 21

|----→ **Creator :** Mr. Rod Jonshon

|---→ Springsoftware comes as set of Jars.

|---→ **Website :** http://spring.io

|--→ **To download software :** Download as zip file from spring.io. spring-framework -4.2.2.RELEASE-dist.zip Spring-Framework-3.2.Release-with-docs.zip

|--→ **To install software :** Extract zip file.

Spring 4.1.6/3.2.6 software installation gives

**libs->** having jarfiles

**docs->** apidocs, reference docs (PDF)

**schemas->** xsd file having schema rules.

Every DTD and XSD file contains rules to construct xml documents.

**DTD->** document type definition

**XSD->** xml schema definition

## Spring Defintion:-

Spring is an **opensource, light weight, loosly coupled, aspect oriented, dependency injection based** java application framework to develop various java applications.

Along with spring software installation we get its source code. This makes spring as open source.

Open source software are not only free software. They also expose their source code along with installation.

Spring is light weight because

- ➔ Spring framework comes as zip file having less size.
- ➔ Spring gives 2 lightweight containers which can activated anywhere in the application. To work with spring container we donot need any application server/ webserver software. We can use specific modules of spring without having any link with other modules.

E.g: We can use spring core module alone or spring core and spring DAO module together without any link with other modules. If degree of dependency is less between modules, then they are called loosly coupled components. E.g TV and remote.

If degree of dependency is more between modules then they are called tightly coupled components.

E.g. Keyboard and System.

In spring framework core module is base module for all modules i.e. we should use every module along with core module. But remaining modules of spring can be used alone without having any dependency with other modules. The degree of dependency is very less between other modules of spring. This makes spring as loosely coupled framework.

AOP is a new methodology of programming that compliments OOP to separate secondary middleware services from primary business logic and allows to link them at runtime dynamically through some configuration. This gives advantages of reusing middleware service logic and flexibility of enabling or disabling middleware services logic on primary logics (like security, transaction and etc…)

## Dependency Lookup:-

If the resource/ class/ Object is searching and getting is dependent values from other resources then it is called dependency lookup. Here the resource pulls the dependent values.

- ➔ The way we get Objects from Jndi registry is called “Dependency Lookup”.
- ➔ The way we get Jdbc connection object from JDBC connection pool is called “Dependency Lookup”.
- ➔ If Student is getting course material only after putting request for it is called dependency lookup.

## Dependency Injection (Inversion of control / IOC):

If underlying server/ container/ framework/ runtime (like JRE) assign values to resources/ class/ objects dynamically at run time then it is called ***dependency injection.***

- ➔ Here dependency values will be pushed to the resource.
- ➔ Eg: JVM calling constructor automatically to assign initial values of object when object is created.
- ➔ Servlet container injects ServletConfig object to our servlet class object by calling init (ServletConfig config) method when servlet Container creates our Servlet class object.
- ➔ Student gets course material the moment he register for course.

## POJO class (Plain Old Java Object Class):-

The java class without any specialities i.e. it is ordinary java class.

While developing this class we need not to follow any serious rule.

The java class that is not extending from technology or framework api class and not implementing technology/ framework api interface is called POJO class.

e.g.

```
class Demo
{……
…….
}
It is POJO class
class Test implements Serializable{
…….....
……..
}
POJO class
```

class Test extends HttpServlet{

……..

…….

}

Test is non POJO class.

Eg.

Class Test implements java.sql.Connection{

……

……

}

“Test” is non POJO

## POJI (Plain Old Java Interface):

It indicates an interface without specialities i.e. an ordinary interface.

The interface that is not extending from technology/ framework api interface is called POJI.

Eg:

interface Test extends Serializable{

…….

……

}

“Test” is POJI.

```
interface Test extends java.rmi.Remote{

…….

……

}
```

"Test" is not POJI.

Eg:

Hibernate, Spring, Struts 2.x etc.

## Java Bean:

It is a java class that is developed with some standards. The standards are

- ➔ Must be public class and should not be final or abstract class.
- ➔ Can be implement Serializable, if needed.
- ➔ Must have direct or indirect 0-parameter constructor.
- ➔ Member variables are called bean properties and they must be taken as private.
- ➔ Every bean property should have setter and getter methods (public).

e.g

```
public class StudentBean{

//bean properties

private int st;

private String name;

//setter and getter

……..

……..

}
```

- ➔ In real time javabean is used as helper class to represent data with multiple values in the form of object and to send those objects from one layer to another layer.

- ➔ JavaBean object is acceptable format to send and receive data across the multiple layers irrespective of technologies or that are used in the layer to develop the logics.
- ➔ The JavaBean class whose object holds input values given by client/ enduser is called VO class.
- ➔ The JavaBean class whose object holds data needed for persistence operation (DAO class) is called as (Business class) BO class.
- ➔ The java bean whose data is shippable over the network is called DTO class.

BO class: Business Object class.

VO class: Value Object class.

DTO class: Data Transfer Object class.

## Bean Class/ Component Class:

It is a java class that contains member variables and methods having business logic manipulating the data member variables.

Eg:

```
class CardProcessor{

private int cardNo;

public int processCard(int cardNo){

……..

……..          //Business Logic

……..

}

public boolean blockCard(int cardNo)

{

……

……..          //Business Logic

……..

}

}
```

The Bean/ component class can extend or implement any class or interface.

## Spring Bean:

- ➔ The java class whose object can be managed by spring container is called SpringBean.
- ➔ SpringBean can be pojo class, java bean class.
- ➔ Bean/ Component class, but its object must be created and manged by spring container.
- ➔ Spring Bean can be user defined class/ pre-defined class/ third party class.
- ➔ We can't take abstract class/interface as SpringBean because it can't create object for abstract class or interface.
- ➔ To make java class as spring beans it must be configure in spring bean configuration file (xml) to make spring container to create and manage that spring Bean class object.

## Important points:

- ➔ Spring Bean can be pojo class? (Yes)
- ➔ Every java bean is pojo class? (Yes)
- ➔ Every pojo class is java Bean? (No)
- ➔ Every spring bean is java Bean? (No)
- ➔ Spring Bean can be a java Bean? (Yes)
- ➔ Spring Bean cannot implement spring api interface? (No)
- ➔ Any component/ bean class can be taken spring as Bean? (Yes)

Spring gives to light weight containers which can be started in any application by just creating object for container classes.

Any java class or application that can manage the life cycle of given resource (class) is called container.

## Spring containers are:

1. **BeanFactory** (Basic container)
   (performs spring bean management and dependency injection)
2. **ApplicationContext** (Advanced Container)
   (internally uses BeanFactory container so can perform advanced operations along with Bean management and dependency injection)
   To activate/ start BeanFactory container we should create object of a class that implements "BeanFactory".

Eg:

BeanFacory Factory=new XmlBeanFactory(Resource object)

|-> Points to spring bean config file (xml file)

|-> Spring containers (light weight) are not replacement/ alternate for servlets, jsp containers (heavy weight)

# Spring Core Module

Base module for other spring module.

Gives two IOC containers/ spring containers.

1. BeanFactory
2. ApplicationContext

When it is used alone we can develop standard applications.

When it is used along other technologies we can develop model layer business logic by taking advantage of dependency injection.

Now talk about spring bean life cycle management and dependency injection.

## Dependency Injection:

If the underlying server or container dynamically assign the dependent values to our resource (class/ object) then it is called dependency injection. There are multiple ways of performing dependency injection:-

a.) Setter injection (Container calls setXxx(-) to inject dependent values)
b.) Constructor injection (container uses parameterized constructor to create bean class object and to inject value to that object)
c.) Interface injection (Aware injection)
d.) Method injection
e.) Lookup Method injection and etc…

## Setter injection sample code:

```
//WishGenerator.java (POJO class)

package com.nt.beans;

import java.util.Date;

public class WishGenerator{

        //bean properties

        private String name;

        private Date date;

        public WishGenerator(){

                System.out.println("WishGenerator: 0-param constructor");}
```

```
//Setter methods for setter injection
        public void setName(String name){
                System.out.println("WishGenerator:setName(-)");
                this.name=name;
        }
        public void setDate(Date date){
                System.out.println("WishGenerator:setDate(-)");
                this.date=date;
        }
        //B. Method
        public String generateWishMsg(){
                return "Good Morning: "+name+"--→"+date;
        }
} //class
```

***applicationContext.xml:-***

```
<beans xmlns=http://.....>

<bean id=”dt” class=”java.util.Date”/>

<bean id=”wsg” class=”com.nt.beans.WishGenerator”/>

<property name=”name” value=”Brijesh”/>

<property name=”date” ref=”dt”/>

</bean>

</beans>
```

---

Any <file-name>.xml can be taken as spring configuration file. But it is recommended to take “applicationContext.xml” in standalone spring applications.

---

# JAVA Interview Questions

## Java Class declaration

1) What is present version of java and initial version of java?

2) How many modifiers in java and how many keywords in java?

3) What is initial name of java and present name of java?

4) Can we have multiple public classes in single source file?

5) Can we create multiple objects for single class?

6) What do you mean by token and literal?

7) What do you mean by identifier?

8) Is it possible to declare multiple public classes in single source file?

9) What is the difference between editor and IDE(integrated development environment)

10) Write the examples of editor and IDE?

11) Define a class?

12) In java program starts form which method and who is calling that method?

13) What are the commands required for compilation and execution?

14) Can we compile multiple source files at a time and is it possible to execute multiple .classes at a time?

15) The compiler understandable file format and JVM understandable file format?

16) What is the difference between JRE and JDK?

17) What is the difference between path and class path?

18) What is the purpose of environmental variables setup?

19) What do you en by open source software?

20) What are operations done at compilation time and execution time?

21) What is the purpose of JVM?

22) JVM platform dependent or independent?

23) In java program execution starts from?

24) How many types of commands in java and what is the purpose of commands?

25) Is it possible to provide multiple spaces in between two tokens?

26) Class contains how many elements based on Brijesh sir class notes?

27) Source file contains how many elements?

28) What are dependent languages and technologies in market on java?

29) Who is generating .class file and .class files generation is based on what?

30) What is .class file contains?

31) What is the purpose of data types and how many data types are present in java?

32) Who is assigning default values to variables?

33) What is the default value of int, char, Boolean, double?

34) Is null is a keyword or not?

35) What do you mean by main class?

36) Is it possible to declare multiple classes with main method?

37) Can I have multiple main methods in single class?

38) What is the default package in java?

39) Can I import same class twice yes-> what happened no -> why ?

40) Do I need to import java.lang package ? yes --->why no--->why?

41) Is empty java source file is valid or not?

42) Is it java file contains more than one class?

43) What is the purpose of variables in java?

44) How many types of variables in java and what are those variables?

45) What is the life time of static variables and where these variables are stored?

46) What is the life time of instance variables and where these variables are stored?

47) What is the life time of local variables and where these variables are stored?

48) For the static members when memory is allocated?

49) Where we declared local variables & instance variables & static variables

50) For the instance members when memory is allocated?

51) For the local variables when memory is allocated?

52) What is the difference between instance variables and static variables?

53) Can we declare instance variables inside the instance methods and static variables inside the static method?

54) If the local variables of methods and class instance variables having same names at that situation how we are represent local variables and how are representing instance variable?

55) What do you mean by method signature?

56) What do you mean by method implementation?

57) How many types of methods in java and how many types of areas in java?

58) What is the purpose of template method?

59) Can we have inner methods in java?

60) One method is able to call how many methods at time?

61) For java methods return type is mandatory or optional?

62) Who will create and destroy stack memory in java?

63) When we will get stackoverFlowError?

64) Is it possible to declare return statement any statement of the method or any specific rule is there?

65) When we will get “variable might not have been initialized” error message?

66) What are the coding conventions of classes and interfaces?

67) What are the coding conventions of methods and variables?

68) What is the default package in java programming?

69) Platform dependent vs platform independent?

70) Is java a object oriented programming language?

71) By using which keyword we are creating object in java?

72) Object creation syntax contains how many parts?

73) How many types of constructors in java?

74) How one constructor is calling another constructor? One constructor is able to call how many constructors at time?

75) What do you mean by instantiation?

76) What is the difference between object instantiation and object initialization?

77) How many ways to create a object in java?

78) What is the purpose of this keyword?

79) Is it possible to use this keyword inside static area?

80) What is the need of converting local variables to instance variables?

81) Is it possible to convert instance variables to local variables yes->how no->why?

82) When we will get compilation error like “call to this must be first statement in constructor”?

83) When we will get compilation error line “cannot find symbol”?

84) What do u mean by operator overloading, is it java supporting operator overloading concept?

85) What is the purpose of scanner class and it is present in which package and introduced in which version?

86) What do you mean by constructor?

87) Who is generating default constructor and at what time?

88) What is the difference between named object and nameless object and write the syntax ?

89) What is object and what is relationship between class and Object?

90) Is it possible to execute default constructor and user defined constructor time?

91) If we are creating object by using new operator at that situation for every object creation how many constructors are executed?

92) What do you mean by object delegation?

93) What is the purpose of instance blocks when it will execute?

94) Inside class it is possible to declare how many instance blocks and what is syntax?

95) What is execution flow of method VS constructor Vs instance blocks Vs static blocks? .

96) When instance blocks and static blocks are executed?

97) What are the new features of java1.5 version VS java1.6 VS java 1.7 VS java 8?

## Flow control statement

1) How many flow control statements in java?

2) What is the purpose of conditional statements?

3) What is the purpose of looping statements?

4) What are the allowed arguments of switch?

5) When we will get compilation error like “possible loss of precision”?

6) Inside the switch case vs. default vs. Break is optional or mandatory?

7) Switch is allowed String argument or not?

8) Inside the switch how many cases are possible and how many default declarations are possible?

9) What is difference between if & if-else & switch?

10) What is the default condition of for loop?

11) Inside for initialization & condition & increment/decrement parts optional or mandatory?

12) When we will get compilation error like “incompatible types”?

13) We are able to use break statements how many places and what are the places?

14) What is the difference between break& continue?

15) What do you mean by transfer statements and what are transfer statements present in java?

16) for (; ;) representing?

17) When we will get compilation error like “unreachable statement “?

18) Is it possible to declare while without condition yes -> what is default condition no -> what is error?

19) What is the difference between while and do-while?

20) While declaring if , if-else , switch curly braces are optional or mandatory?

## OOPS

1) What are the main building blocks of oops?

2) What do you mean by inheritance?

3) How to achieve inheritance concept and inheritance is also known as?

4) How many types of inheritance in java and how many types of inheritance not supported by java?

5) How to prevent inheritance concept?

6) What is the purpose of extends keyword?

7) What do you mean by cyclic inheritance java supporting or not?

8) What is the difference between child class and parent class?

9) What is the root class for all java classes?

10) Inside the constructor if we are not providing this() and super() keyword the compiler generated which type of super keyword?

11) How to call super class constructors?

12) Is it possible to use both super and this keyword inside the method?

13) Is it possible to use both super and this keyword inside the constructor?

14) If the child class and parent class contains same variable name that situation how to call parent class variable in child class?

15) One class able to extends how many classes at a time?

16) If we are extending the your class will become parent class if we are not extending what is the parent class?

17) What do you mean by aggregation and what is the difference between aggregation and inheritance?

18) What do you mean by aggregation and composition and Association?

19) Aggregation is also known as?

20) How many objects are created ?

a. MyClassHero c1,c2;

C1 = New MyClassHero();

21) What is the root class for all java classes?

22) Which approach is recommended to create object either parent class object or child class object?

23) Except one class all class contains parent class in java what is that except class?

24) What is the purpose of instance of keyword in java?

25) What do you mean by polymorphism?

26) What do you mean by method overloading and method overriding?

27) How many types of overloading in java?

28) Is it possible to override variable in java?

29) What do you mean by constructor overloading?

30) What are rules must fallow while performing method overriding?

31) When we will get compilation error like “overridden method is final”?

32) What is the purpose of final modifier java?

33) Is it possible to override static methods yes-> how no-> why?

34) Parent class reference variable is able to hold child class object?

35) How many types of polymorphism in java?

36) What do you mean by dynamic method dispatch?

37) The applicable modifiers for local variables?

38) Is it all methods present in final class is always final and is it all variables present final class is always final?

39) If Parent class is holding child class object then by using that we are able to call only overridden methods of child class but how to call direct methods of child class?

40) Object class contains how many methods?

41) When we will get compilation error like “con not inherit from final parent”?

42) How many types of type casting in java?

43) What do you mean by co-variant return types?

44) What do u mean by method hiding?

45) What do you mean by abstraction?

46) How many types of classes in java?

47) Normal class is also known as ?

48) What is the difference between normal method and abstract method?

49) What is the difference between normal class and abstract class?

50) Is it possible to create a object for abstract class?

51) What do you mean by abstract variable?

52) Is it possible to override non-abstract method as a abstract method?

53) Is it possible to declare main method inside the abstract class or not?

54) What is the purpose of abstract modifier in java?

55) How to prevent object creation in java?

56) What is the definition of abstract class?

57) In java is it abstract class reference variable is able to hold child class object?

58) What do you mean by encapsulation?

59) What do you mean by tightly encapsulated class?

60) What do you mean accessor method and mutator method ?

61) How many ways area there to set some values to class properties?

62) Can we overload method?

63) Can we inherit main method in child class?

64) In java main method is called by ?

65) The applicable modifiers on main method?

66) While declaring main method public static modifiers order mandatory or optional?

67) What is the argument of main method?

68) What is the return type of main method?

69) What are the mandatory modifiers for main method and optional modifiers of main method?

70) Why main method is static?

71) What do you by command line arguments?

72) Is it possible to pass command line arguments with space symbol no-> good yes-> how ?

73) What is the purpose of strictfp classes?

74) What is the purpose of strictfp modifier?

75) What is the purpose of native modifier?

76) What do you mean by native method and it also known as?

77) What do you mean by javaBean class?

78) The javabean class is also known as?

79) Applicable modifiers on local variables?

80) What is the execution process of constructors if two classes are there in inheritance relationship?

81) What is the execution process of instance blocks if two classes are there in inheritance relationship?

82) What is the execution process of static blocks if two classes are there in inheritance relationship?

83) What is the purpose of instanceof operator in java & what is the return-type?

84) If we are using instanceof both reference-variable & class-name must have some relationship otherwise compiler generated error message is what?

## Packages

1. What do you mean by package and what it contains?

2. What is the difference between user defined package and predefined package?

3. What are coding conventions must fallow while declaring user defined package names?

4. Is it possible to declare motile packages in single source file?

5. What do you mean by import?

6. What is the location of predefined packages in our system?

7. How many types of imports present in java explain it?

8. How to import individual class and all classes of packages and which one is recommended?

9. What do you mean by static import?

10. What is the difference between normal and static import?

11. Is it possible to import multiple packages in single source file?

12. Is it possible to declare multiple packages in single source file?

13. I am importing two packages, both packages contains one class with same name at that situation how to create object of two package classes?

14. If we are importing root package at that situation is it possible to use sub package classes in our applications?

15. What is difference between main package and sub package?

16. If source file contains package statement then by using which command we are compiling that source file?

17. What do you mean by fully qualified name of class?

18. What is the public modifier?

19. What is the default modifier in java?

20. What is the public access and default access?

21. What is private access and protected access?

22. What is the difference between public methods and default method?

23. What is the difference between private method and protected method?

24. What is most restricted modifier in java?

25. What is most accessible modifier in java?

## Exception Handling

1. What do you mean by Exception?

2. How many types of exceptions in java?

3. What is the difference between Exception and error?

4. What is the difference between checked Exception and un-checked Exception?

5. Checked exceptions are caused by?

6. Unchecked exceptions are caused by?

7. Errors are caused by?

8. Is it possible to handle Errors in java?

9. What the difference is between partially checked and fully checked Exception?

10. What do you mean by exception handling?

11. How many ways are there to handle the exception?

12. What is the root class of Exception handling?

13. Can you please write some of checked and un-checked exceptions in java?

14. What are the keywords present in Exception handling?

15. What is the purpose of try block?

16. In java is it possible to write try with out catch or not?

17. What is the purpose catch block?

18. What is the difference between try-catch?

19. Is it possible to write normal code in between try-catch blocks?

20. What are the methods used to print exception messages?

21. What is the purpose of printStackTrace( ) method?

22. What is the difference between printStackTrace( ) & getMessage()?

23. What is the purpose of finally block?

24. If the exception raised in catch block what happened?

25. Independent try blocks are allowed or not allowed?

26. Once the control is out of try , is it remaining statements of try block is executed?

27. Try-catch , try-catch-catch , catch-catch , catch-try how many combinations are valid?

28. Try-catch-finally , try-finally ,catch-finally , catch-catch-finally how many combinations are valid?

29. Is possible to write code in between try-catch-finally blocks?

30. Is it possible to write independent catch blocks?

31. Is it possible to write independent finally block?

32. What is the difference between try-catch –finally?

33. What is the execution flow of try-catch?

34. If the exception raised in finally block what happened?

35. What are the situations finally block is executed?

36. What are the situations finally block is not executed?

37. What is the purpose of throws keyword?

38. What is the difference between try-catch blocks and throws keyword?

39. What do you mean by default exception handler and what is the purpose of default exception handler?

40. How to delegate responsibility of exception handling calling method to caller method?

41. What is the purpose of throw keyword?

42. If we are writing the code after throw keyword usage then what happened?

43. What is the difference between throw and throws keyword?

44. How to create user defined checked exceptions?

45. How to create user defined un-checked exceptions?

46. Where we placed clean-up code like resource release, databasecloseting inside the try or catch or finally and why ?

47. Write the code of ArithmeticException?

48. Write the code of NullPointerException?

49. Write the code of ArrayIndexOutOfBoundsException & StringIndexOutOfBoundsException?

50. Write the code of IllegalThreadStateException?

51. When we will get InputMisMatchException?

52. When we will get IllegalArgumantException?

53. When we will get ClassCastException?

54. When we will get OutOfMemoryError?

55. When we will get compilation error like “unreportedException must be catch”?

56. When we will get compilation error like “Exception XXXException has already been caught”?

57. When we will get compilation error like “try without catch or finally”?

58. How many approaches are there to create user defined unchecked exceptions and un-checked exceptions?

59. What do you mean by exception re-throwing?

60. How to create object of user defined exceptions?

61. How to handover user created exception objects to JVM?

62. What is the difference user defined checked and unchecked Exceptions?

63. Is it possible to handle different exceptions by using single catch block yes-->how no-> why?

## Interfaces

1. What do you mean by interface how to declare interfaces in java?

2. Interfaces allows normal methods or abstract methods or both?

3. For the interfaces compiler generates .class files or not?

4. Interface is also known as?

5. What is the abstract method?

6. By default modifiers of interface methods?

7. What is the purpose of implements keyword?

8. Is it possible to declare variables in interface ?

9. Can abstract class have constructor ? can interface have constructor?

10. What must a class do to implement interface?

11. What do you by implementation class?

12. Is it possible to create object of interfaces?

13. What do you mean by abstract class?

14. When we will get compilation error like “attempting to assign weaker access privileges”?

15. What is the difference between abstract class and interface?

16. What do you mean by helper class?

17. what is the difference between classes and interfaces?

18. The interface reference variable is able to hold implementation class objects or not? a. Interface-name reference-variable = new implementation class object(); valid or invalid

19. What is the real-time usage of interfaces?

20. what is the limitation of interfaces how to overcome that limitation?

21. What do you mean by adaptor class?

22. What is the difference between adaptor class interfaces?

23. Is it possible to create user defined adaptor classes?

24. Tell me some of the adaptor classes?

25. What do you mean by marker interface and it is also known as?

26. Tell me some of the marker interfaces?

27. What are the advantages of marker interfaces?

28. Is it possible to create user defined marker interfaces /

29. What do you mean nested interface?

## String Manipulation

1) How many ways to create a String object & StringBuffer object?

2) What is the difference between

a. String str=”softpro”;

b. String str = new String(“softpro”);

3) equals() method present in which class?

4) What is purpose of String class equals() method.

5) What is the difference between equals() and == operator?

6) What is the difference between by immutability & immutability?

7) Can you please tell me some of the immutable classes and mutable classes?

8) String & StringBuffer & StringBuilder & StringTokenizer presented package names?

9) What is the purpose of String class equals() & StringBuffer class equals()?

10) What is the purpose of StringTokenizer nd this class functionality replaced method name?

11) How to reverse String class content?

12) What is the purpose of trim?

13) Is it possible to create StringBuffer object by passing String object as a argument?

14) What is the difference between concat() method & append()?

15) What is the purpose of concat() and toString()?

16) What is the difference between StringBuffer and StringBuilder?

17) What is the difference between String and StringBuffer?

18) What is the difference between compareTo() vs eqauls()?

19) What is the purpose of contains() method?

20) What is the difference between length vs length()?

21) What is the default capacity of StringBuffer?

22) What do you mean by factory method?

23) Concat() method is a factory method or not?

24) What is the difference between heap memory and String constant pool memory?

25) String is a final class or not?

26) StringBuilder and StringTokenizer introduced in which versions?

27) What do you mean by legacy class & can you please give me one example of legacy class?

28) How to apply StringBuffer class methods on String class Object content?

29) When we use String & StringBuffer & String

30) What do you mean by cloneaing and use of cloning?

31) Who many types of cloneaing in java?

32) What do you mean by cloneable interface present in which package and what is the purpose?

33) What do you mean by marker interface and Cloneable is a marker interface or not?

34) How to create duplicate object in java(by using which method)?

## Collections

1) What is the main objective of collections?

2) What are the advantages of collections over arrays?

3) Collection frame work classes are present in which package?

4) What is the root interface of collections?

5) List out implementation classes of List interface?

6) List out implementation classes of set interface?

7) List out implementation classes of map interface?

8) What is the difference between heterogeneous and homogeneous data?

9) What do you mean by legacy class can you please tell me some of the legacy classes present in collection framework?

10) What are the characteristics of collection classe?

11) What is the purpose of generic version of collection classes?

12) What is the difference between general version of ArrayList and generic version of ArrayList?

13) What is purpose of generic version of ArrayList & arrays?

14) How to get Array by using ArrayList?

15) What is the difference betweenArrayList and LinkedList?

16) How to decide when to use ArrayList and when to use LinkedList?

17) What is the difference between ArrayList & vector?

18) How can ArrayList be synchronized without using vector?

19) Arrays are already used to hold homogeneous data but what is the purpose of generic version of Collection classes?

20) What is the purpose of RandomAccess interface and it is marker interface or not?

21) What do you mean by cursor and how many cursors present in java?

22) How many ways are there to retrieve objects from collections classes what are those?

23) What is the purpose of Enumeration cursor and how to get that cursor object?

24) By using how many cursors we are able to retrieve the objects both forward backward direction and what are the cursors?

25) What is the purpose of Iterator and how to get Iterator Object?

26) What is the purpose of ListIterator and how to get that object?

27) What is the difference between Enumeration vs Iterator Vs ListIterator?

28) We are able to retrieve objects from collection classes by using cursors and for-each loop what is the difference?

29) All collection classes are commonly implemented some interfaces what are those interfaces?

30) What is the difference between HashSet & linkedHashSet?

31) all most all collection classes are allowed heterogeneous data but some collection classes are not allowed can you please list out the classes?

32) What is the purpose of TreeSet class?

33) What is the difference between Set & List interface?

34) What is the purpose of Map interface?

35) What do you mean by entry.

36) What is the difference between HashMap & LinkedHashMap?

37) What is the difference between comparable vs Comparator interface?

38) What is the difference between TreeSet andTtreeMap?

39) What is the difference between HashTable and Properties file key=value pairs?

40) What do you mean by properties file and what are the advantages of properties file?

41) Properties class present in which package?

42) What is the difference between collection & collections?

## Nested Classes

1) What are the advantages of inner classes?

2) How many types of nested class?

3) How many types of inner classes?

4) What do you by static inner classes?

5) The inner class is able to access outer class private properties or not?

6) The outer class is able to access inner classes properties& methods or not? 7) How to create object inner class and outer class?

a. Class Outer {

class Inner{ }

}

8) For the inner classes compiler generates .class files or not? If generates write the name of above inner class .class file name ?

9) The outer class object is able to call inner class properties & methods or not?

10) The inner class object is able to call outer class properties and methods or not?

11) What is the difference between normal inner classes and static inner classes?

12) What do you mean by anonymous inner classes?

13) What do you mean by method local inner classes?

14) Is it possible to create inner class object without outer class object?

15) Java supports inner method concept or not ?

16) Is it possible to declare main method inside inner classes?

17) Is it possible to declare constructors inside inner classes?

18) If outer class variables and inner class variables are having same name then hoe to represent outer class variables and how to represent inner class variables?

19) Is it possible to declare same method in both inner class and outer class?

20) Is it possible to declare main method inside outer classes?

## Multithreading

1. What do you mean by Thread?

2. What do you mean by single threaded model?

3. What is the difference single threaded model and multithreaded model?

4. What do you mean by main thread and what is the importance?

5. What is the difference between process and thread?

6. How many ways are there to create thread which one prefer?

7. Thread class& Runnable interface present in which package?

8. Runnable interface is marker interface or not?

9. What is the difference between t.start() & t.run() methods where t is object of Thread class?

10. How to start the thread?

11. What are the life cycle methods of thread?

12. Run() method present in class/interface ?Is it possible to override run() method or not?

13. Is it possible to override start method or not?

14. What is the purpose of thread scheduler?

15. Thread Scheduler fallows which algorithm?

16. What is purpose of thread priority?

17. What is purpose of sleep() & isAlive() & isDemon() & join() & getId() & activeCount() methods?

18. Jvm creates stack memory one per Thread or all threads only one stack?

19. What is the thread priority range & how to set priority and how to get priority?

20. What is the default name of user defined thread and main thread? And how to set the name and how to get the name?

21. What is the default priority of main thread?

22. Which approach is best approach to create a thread?

23. What is the difference between synchronized method and non-synchronized method? 24. What is the purpose of synchronized modifier?

25. What is the difference between synchronized method and non synchronized method?

26. What do you mean by demon thread tell me some examples?

27. what is the purpose of volatile modifier?

28. What is the difference between synchronized method and synchronized block?

29. Wait() notify() notifyAll() methods are present in which class?

30. When we will get Exception like “IllegalThreadStateException” ?

31. When we will get Exception like “IlleglArgumentException” ?

32. If two threads are having same priority then who decides thread execution?

33. How two threads are communicate each other?

34. What is race condition?

35. How to check whether the thread is demon or not? Main thread is demon or not?

36. How a thread can interrupt another thread?

37. Explain about wait() motify() notifyAll()?

38. Once we create thread what is the default priority?

39. What is the max priority & min priority & norm priority?

40. What is the difference between preemptive scheduling vs time slicing?

# Thank You