

(Similar to 03/07/2015
which not attend)
diff. class

Differences between Java and Others:-

- 1) C and C++ are static Programming languages but Java is dynamic programming language.
- Static Programming language:- If any programming language allows memory allocation for primitive data type at compilation time then that programming language is called as static program language.
Ex:- C and C++.

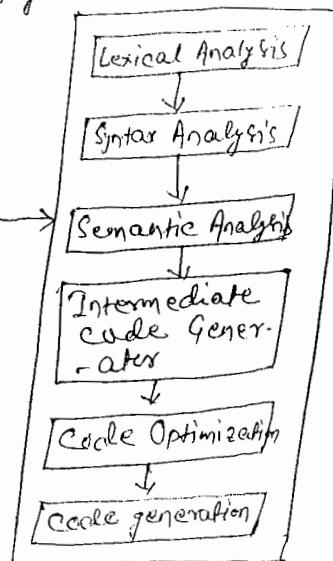
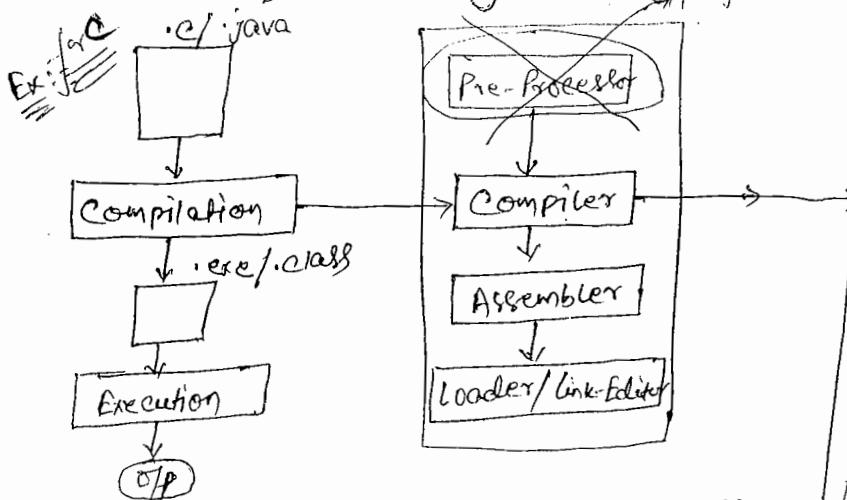
→ Dynamic programming language:-

- If any programming language allows memory allocation at runtime, not at compilation time then that programming language is called as dynamic Programming language.

Ex:- Java.

NOTE:- In java applications, memory is allocated for primitive data types at the time of creating Objects, in java applications Objects are created at runtime.

- 2) Pre-Processor is required in C and C++, but Pre-processor is not required in java.



Ex of java:-
packages.

Java.io
Java.Util
Java.Sql

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

→ In C and C++, the complete predefined library is provided in the form of header files.

Ex:- stdio.h
conio.h
math.h

In C and C++ applications, if we want to include header files then we have to use #include<> statements.

If we compile C and C++ programs, pre-processor will recognise #include<> statements and perform the following actions:-

- 1) Pre-Processor will take all the specified header files name from #include<> statements.
- 2) Pre-Processor will search for the specified header files at C and C++ predefined library.
- 3) If the required header files are identified then pre-processor will load header files content into memory.

NOTE: Loading Predefined library at the time of compilation is called as Static loading.

In C and C++, header files are Mandatory, so #include statements are mandatory, so, that Pre-processor is required in C and C++.

(2) 15/07/2015 2

In java, the complete predefined library is provided in the form of packages. (see the fig on next page for java api).

Ex:- `java.io` ✓
`java.util` ✓
`java.sql` ✓

To use java applications to include predefined library we have to use "import" statements.

Ex:- `import java.io.*;` ✓
`import java.util.*;` ✓
`import java.sql.*;` ✓

When we compile java file, compiler will recognize import statements and compiler will perform the following actions.

- 1) Compiler will take package names from import statements.
- 2) Compiler will check whether the specified package are existed or not at java predefined library.
- 3) If the specified packages are not existed then compiler will rise an error like "package xxx does not exist".
- 4) If the specified packages are existed then compiler will not rise any error and compiler will not load any class or interface at compilation time.

When we execute java program, when JVM encounters any class or interface then Only JVM will load the respective class or interface by bytecode to the memory.

NOTE:- Loading classes or interfaces by bytecode at runtime is called as "Dynamic loading".

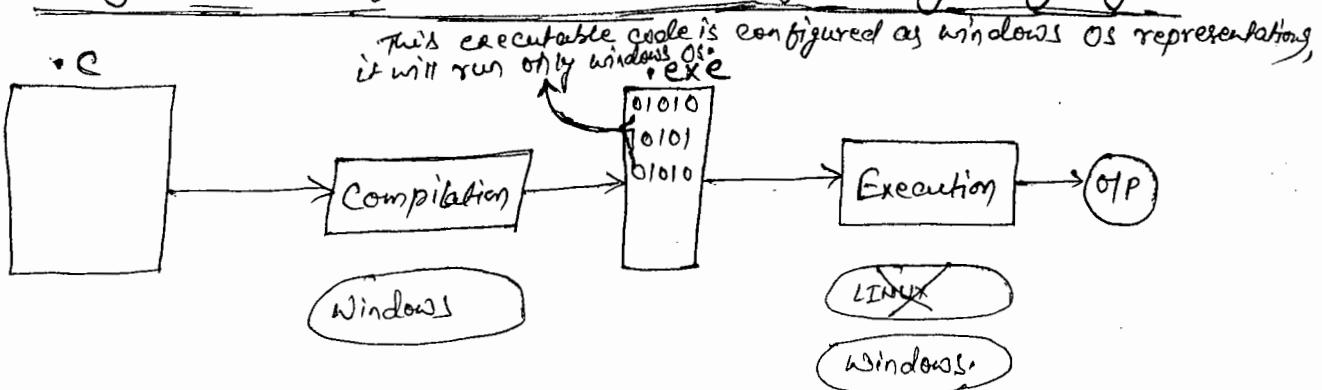
In java, headers files are not available, so #include <> statements are not available, so that, Pre-processor is not required.

Q. What are differences between #include <> statement and import statement.

Ans.

- 2) #include < > statements are available upto C and C++.
import statements are available upto Java.
- 2) #include < > statements can be used to include the predefined library available in the form of header files.
import statement can be used to include the predefined library available in the form of packages.
- 3) #include < > statements are recognized by pre-processor.
import statements are recognized by compiler and JVM.
- 4) #include < > statements are providing static loading.
Import statements are providing dynamic loading.
- 5) By Using single #include < > statements we are able to include
more than one header file.
By Using single "import" statement we are able to include
more than one class or interface of the same package
by Using ~~dot~~ notation.

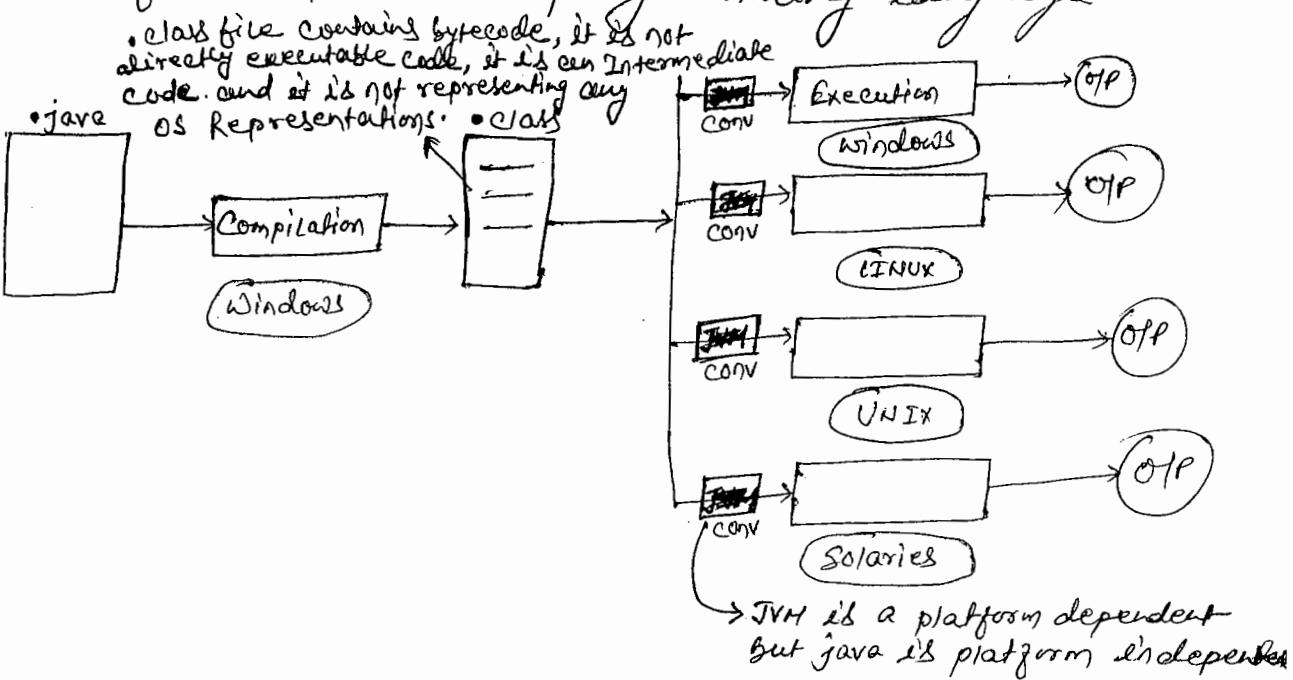
3. C and C++ are platform dependent programming languages
But Java is platform independent programming language:-



If any programming language allows its applications to compile and to execute on the same OS then that programming language is called as platform dependent programming language.

Ex:- C and C++.

If we compile C program on windows operating system then compiler will generate .exe file as per windows representations. In this context, to execute windows represented .exe file we must use the same windows OS at runtime, this nature of C programming language is platform dependent programming language.



If any programming language allows its applications to compile on one OS and to execute on other OS then that programming language is called as platform independent programming language.

Ex:- Java.

If we compile java program on windows OS then compiler will generate .class file, which contains bytecode, it is not directly executable code, It is intermediate code and it is not representing any OS representations including windows OS.

In the above context, to execute java programs in any other Operating Systems then the respective bytecode must represent local OS representations.

In this context, to make java as platform independent programming language and to execute java program in different OS, Sun Microsystem has provided a converter in the form of JVM.

In java application execution, JVM is responsible to convert neutral bytecode into local Operating System representations and to execute java programs.

[NOTE:- Java is platform independent programming language due to availability of JVMs but JVMs is platform dependent.]

17/07/2015

Programs are NOT Possible in Java ??

Q. What are the diff. between .exe file and .class file?

Ans. i) .exe file is upto C and C++.
ii) .class file is upto Java.

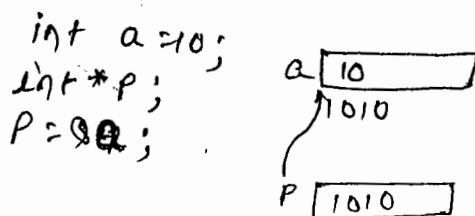
- 2) .exe file containing directly executable code.
- .class file contains bytecode, it is not directly executable code, it is an intermediate code.
- 3) .exe file is platform dependent.
- .class file is platform independent.

(4)

4) Pointers are not possible in java:-

Pointer is a variable, it able to stores address ~~location~~ directly.

Pointer Variables are recognized and initialized at the time of compilation.



Pointers are not possible in java, because,

1) Pointer Variable must require static memory allocation, but java is following dynamic memory allocation.

2) Pointer Variables are supported by static programming languages, but java is dynamic programming language.

3) Pointer variables are suitable in platform dependent programming languages, but java is platform independent programming language.

4) Pointer Variables will reduce security for the application data but java is very good secure programming language, it has to provide very good security for the application data.

5) Pointers will increase confusion to the developers but java is a simple programming language, it should not provide confusion to the developers.

Q. What are the differences between pointer variables and reference Variable ?

1) Pointer Variables are Variable upto C and C++.
Reference Variables are available upto java.

2) Pointer Variables are recognized and initialized at the time of compilation.
Reference Variables are recognized and initialized at runtime.

3) Pointer Variables are able to refer a block of memory by storing its address location..
Reference Variables are able to refer a block of memory [Object] by Storing object reference Value.

NOTE:- Object reference Value is hexadecimal form of hashcode, where hashcode is an unique identity provided by Heap manager in the form of random integer value for each and every object.

5) C and C++ Using call by Value and call by reference
But java is Using call by Value parameter passing mechanism:-

In any programming language, if we pass primitive data as parameters to the methods then the parameter passing mechanism is call by value. If we pass address locations as parameters to the methods then the parameter passing mechanism is call by reference.

In case of C and C++, if we pass pointer variable as parameters to the methods then the parameter passing mechanism is call by reference, because pointer variable able to store address locations directly.

In case of java, if we pass reference variables as parameters to the methods then the parameter passing mechanism is call by value Only, not call by reference, because reference variables are not stored address locations.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

⑦ Destructor are not required in java:-

In object Oriented Programming languages, it is convention to store data in the form of Objects. To store data in the form of objects, first we have to create Object and at the end of the program we have to destroy Objects.

In all the object Oriented programming languages, to create Object we have to use constructors and to destroy objects we have to use destructors.

In case of java, to destroy objects automatically java has provided an implicit tool in the form of "Garbage Collector".

Due to availability of Garbage Collector in JAVA, java developers are not required to use destructors.

In case of C++, developers must be destructors to destroy Objects, because in C++ Garbage collector is not available.

6) Multiple inheritance is not possible in java:-

The process of getting variables and methods from one class (super class) to another class (sub class) is called as inheritance.

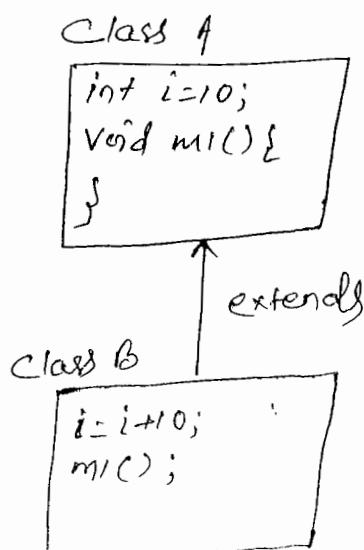
The main objective of inheritance is to improve code reusability.

At basic level inheritance, there are two types of inheritance:

- 1) Single Inheritance
- 2) Multiple Inheritance.

1) Single Inheritance:-

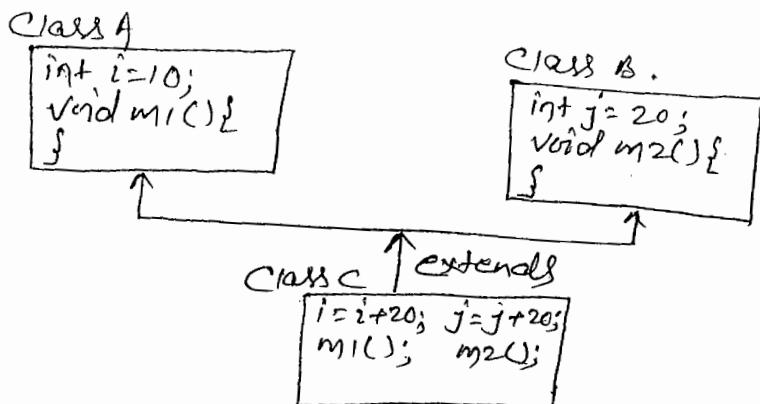
The process of getting variables and methods from only one super class to one or more sub classes is called as Single Inheritance.



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

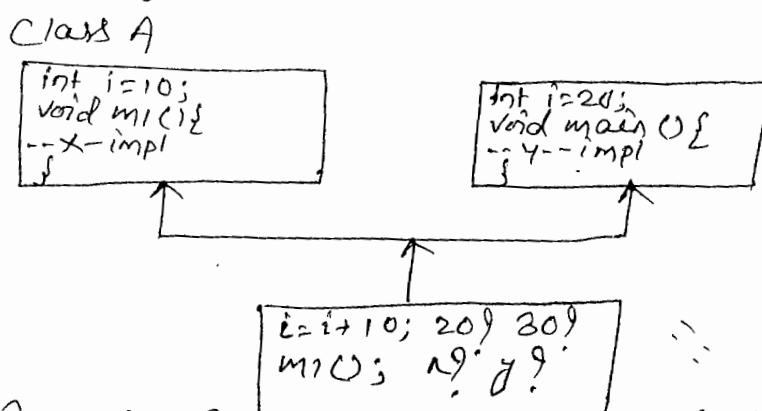
2) Multiple inheritance:-

The process of getting variables and methods from more than one super class to one or more no. of sub classes is called as Multiple Inheritance.



In the above multiple inheritance, if we declare same variable with different values and same method with different implementations and if we access that variable and method in the respective sub class then which super class variable would be accessed and which super class method would be accessed is a confusion state.

Java is a simple programming language it should not allow confusion Oriented feature. So, that Java is not allowing multiple inheritance.



[NOTE:- To restrict multiple inheritance in Java applications, Java has prepared 'extends' keyword to allow only one super class, not to allow more than one super class].

8) Operator Overloading is not available in Java:

If we declare any operator with more than one functionality then it is called as ~~overloaded operators~~ ~~operator Overloading~~ and this process is called as ~~operator Overloading~~.

Ex- int a=10;
int b=20;
int c=a+b; + is for arithmetic addition

System.out.println(c); ----> 30

String str1 = "abc"

String str2 = "def"

String str3 = str1+str2; + is for string concatenation -

System.out.println(str3); -----> abcdef

8) Operator Overloading is not available in Java because, i) Operator Overloading is rarely used feature in application development.

ii) Operator Overloading will increase confusion to the developers when we declare more no. of operators and with more no. of functionalities.

NOTE:- java has declared some Operators like +, *, %, ... as predefined Overloaded Operators as per its implicit requirement. But java has not provided any Option to the developers to perform Overloading explicitly. 10/07/2015

- ⑨ C and C++ are following Call by value and call by Reference parameter passing mechanisms but java is following Call by value parameter passing mechanism.

In any programming language, if we pass primitive data as parameters to methods then the parameter passing mechanism is call by value.

In any programming language, if we pass address locations as parameters to the methods then the parameter passing mechanism is call by reference.

In case of C and C++ applications, if we pass pointer variables as parameters to the methods then the parameter passing mechanism is call by reference, because pointer variables are able to store address locations directly.

In case of java we pass reference variables as parameters to the methods then the parameter passing mechanism is call by value Only, not call by reference, because reference variables are not storing address locations.

- 3) In C and C++, 2 bytes of memory for integers and 1 byte of memory for characters are assigned, but, in java 4 bytes of memory for integers and 2 bytes of memory for characters are assigned.

byte	1 byte
short	2 bytes
int	4 bytes
long	8 bytes
float	4 bytes
double	8 bytes
char	2 bytes
boolean	1 bit

In case of C and C++, memory allocation for primitive data type is not fixed, It is variable from one Operating System to another Operating System.

In case of Java, memory allocation for primitive data type is fixed irrespective of the Operating System which we used.

Q. In C and C++, One byte of memory is sufficient to store characters but what is the requirement of Java to assign two bytes of memory for characters?

A. In C and C++ applications all the characters are represented in the form of ASCII values, to store any ASCII value one byte of memory is sufficient.

In case of Java, all the characters are represented in the form of UNICODE values, to store any UNICODE values 2 bytes of memory is required, due to this reason Java must assign 2 bytes of memory to store characters data.

[NOTE]- UNICODE is one of the character representation, it able to represent all the alphabet from all the natural languages and it able to provide very good Internationalization support.

NOTE- Designing Java applications w.r.t the local conventions is called as Internationalization. If we provide internationalization service in Java applications then we can provide input to the Java programs in our local languages like Hindi, Telugu, and we are able to get output from Java program in the form of our local languages.

Java Features:-

To expose the nature of java programming language has provided the following features.

- 1) Simple
- 2) Object Oriented
- 3) Platform Independent
- 4) Arch neutral (Architectural Neutral).
- 5) Portable
- 6) Robust
- 7) Secure
- 8) Dynamic
- 9) Multi Threaded
- 10) Distributed
- 11) Interpreted
- 12) High performance

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

1) Simple:-

- java is a simple Programming language, because
- 1) Java applications will take less memory and less execution time.
 - 2) Java has removed all the confusion Oriented features like pointers, multiple inheritance, Operator Overloading, ...
 - 3) Java is Using all the Simplified syntax from C and C++.

2) Object Oriented:-

java is an Object Oriented Programming language, because java is able to store data in the form of objects. [Java is partially Object Oriented. But compared with other's java is more Object Oriented. java is not a 100% Object Oriented but 99.9% is Object Oriented.]

3) Platform Independent :-

java is platform independent programming language, because, java is able to allow its applications to compile on one operating system and to execute on another operating system.

4) Architectural Neutral (Arch. Neutral) :-

java is an arch. neutral programming language, because, java is able to allow its applications to compile on one H/W arch and to execute on another H/W arch.

5) Portable :-

java is a portable programming language, because, java is able to execute applications on all the operating systems and on all the H/W systems.

6) Robust :-

If any programming language is very good at memory management system and having very good exception handling mechanisms then that programming language is called as robust programming language.

java is robust programming language, because,

1) Java is having ^{very} good memory management S/m in the form of ~~heap~~ memory management S/m, a dynamic memory management S/m, it allocates and deallocates memory for the objects at the runtime as per the application requirement.

2) Java is having very good exception handling mechanism due to availability of very good pre-defined library to represent and handle all the exceptional situations.

7) Secure:-

Java Authentication and Authorization Service [JAAS].

- Java is very good secure programming language, because,
- 1) Java has provided a separate component in the form of "Security Manager" inside JVM to implement implicit security.
 - 2) Java has provided a separate service in the form of JAAS to provide web security.
 - 3) Java has provided predefined implementation for all well known network security algorithms in order to provide network security.

[NOTE:- Java has removed pointers feature in order to make java as secure programming language.]

8) Dynamic:-

Java is a dynamic programming language because, java is able to allow memory allocation for primitive data types at runtime, not at compilation time.

9) Multi Threading:-

Java is Multi threaded programming language because, java is able to provide very good environment to create and execute more than one thread at a time.

Thread is a flow of execution to perform a particular task.

10) Distributed:-

If we design any java application without using Client-Server architecture then the application is called as Standalone Application. To design Standalone applications we are able to use core libraries.

If we design any java application on the basis of client server arch then that app is called as distributed application.

To design distributed application java has provided the following technology.

Socket programming.

RMI

CORBA

EJBs

Web Services.

- - - - .

Due to the availability of these technologies to design distributed applications, Java is becoming as Distributed Programming language.

1) Interpretive:-

java is both Compilative and Interpretive programming language, because

- 1) To check developers mistakes in java programs we have to use java Compilers.
- 2) To execute java programs, we have to use an interpreter inside JVM.

12) High Performance:-

Java is high performance programming language due to the availability of the java features like platform independent, Arch Neutral, Portable, Robust, Secure, Dynamic.

Java Naming Conventions:-

1) String
StringBuffer
InputStreamReader

Java is Strictly case sensitive Programming language, where in java applications there is a separate reorganization for lower case letter's and Upper case letters.

To use lower case letters and upper case letters Separately in java applications, java has provided the following conventions.

1) All java class names, abstract class names, interface names and enum names must starts with Upper case letters and the subsequent symbols must also be upper case letters.

Ex:-

2) All java Variables must start with lower case letters but the sub sequent symbol must be Upper Case letters.

Ex:- in, out, err

pageContext, bodyContent

3) All java methods must start with lower case letters but the sub sequent symbols must be Upper Case letters.

Ex:-

Contact(-)

forName(-)

getInputStream(-)

4) All java constant variables must be provided in upper case letters.

Ex:-

MIN-PRIORITY

MAX-PRIORITY

NORM-PRIORITY

5) All java package names must be provided in lower case letters.

Ex:-

java.io

java.awt.event

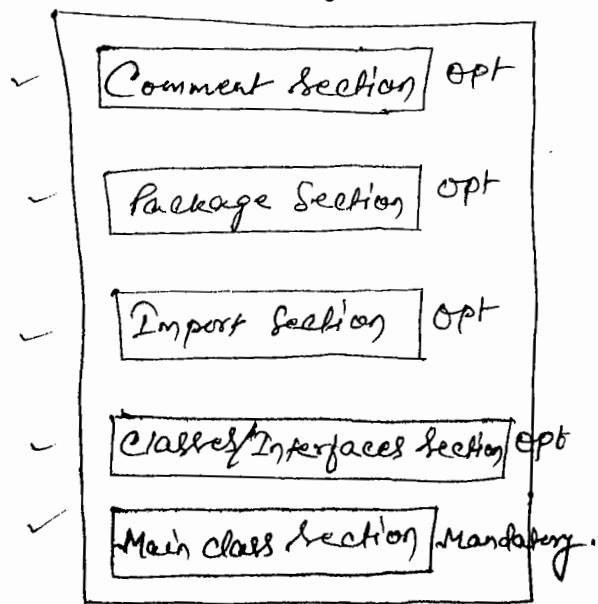
javax.servlet.jsp.tagext

[NOTE: All the above conventions are mandatory for predefined library, which are optional for user defined library but suggestible.]

④ Java Programming Format:-

16/07/2015

To design basic java application within a single java file we have to use the following structure.



⇒ Comment Section:-

As part of java application development before starting implementation part we have to provide some description about our implementation, which includes author name, project details, module details, client details, ...

To provide the above description in java files we have to use comment section, where in comment section we have to use comments.

In java there are three types of comments.

1) Single line Comment:-

If able to allow the description with in a single line.

Syntax:-

//----- Description -----

2) Multiline Comment:-

If able to allow the description in more than one line.

Syntax:-

/*-----
----- description -----

3) Documentation Comment:-

If able to allow the description in more than one page.

Syntax:-

/*
*-----
*-----
-----Description.
*-----
*-----
*/

In general we will utilize documentation comment to prepare API documentation for our java programs.

NOTE:- Getting all programming elements declarative information in the form of a document is called as API documentation. Using documentation comments to prepare API documentation is not suggestible, because it may take lot of time to prepare API documentation.

In the above context, to simplify API documentation preparation, java has provided an implicit tool in the form of "javadoc".

Example:-

Q) C:\abc\Employee.java.

```
*  
public class Employee  
{  
    public String eid;  
    public String ename;  
    public float esal;  
    public float eaddr;  
    public Employee()  
    {  
    }  
    public Employee(String eid)  
    {  
    }  
    public Employee (String eid, String ename, float esal, String eaddr)  
    {  
    }  
    public void search (String eid)  
    {  
    }  
    public void delete (String eid)  
    {  
    }  
}
```

Cont...

On command prompt

C:\abe> javadoc Employee.java

To describe metadata in java applications, JDK5.0 version has provided a new feature in the form of "Annotations".

Q. In java applications, to describe metadata we have already comments then what is the requirement to go for Annotations?

A. In java applications, to provide description if we use comments then "Lexical Analyser" will remove comments and their metadata from java application as part of compilation. In this case metadata ~~data~~ which we specified in java appn is not available in .class files and not available at RUNTIME of our java application.

As per the requirement, if we want to bring metadata upto runtime, upto .class files and upto .java files we have to use "Annotations" instead of comments.

Q. To provide metadata at runtime of our application we are able to use XML documents then what is the requirement to use "Annotations"?

A. To provide metadata at runtime of java application if we use XML documents then we are able to get the following problems.

- 1) We must learn XML tech.
- 2) We have to check every time whether XML documents are located properly or not.
- 3) We have to check every time whether XML documents are formatted properly or not.
- 4) We have to check every time whether we are using right parsing mechanisms or not.

To overcome all the above problems we need a Java alternative that is "Annotations".

17/07/2015

Package Section:-

Package is a collection of related classes and interfaces as a single unit.

Package is a folder containing .class files representing related classes and interfaces.

In java applications, packages will provide the following advantages.

- 1) Modularity
- 2) Abstraction
- 3) Security
- 4) Sharability
- 5) Reusability.

There are two types of packages in java:-

1) Predefined packages:

These packages are defined by java programming language and provided along with java software.

Ex:- java.io
java.util
java.awt
java.sql

2) User defined Packages:-

These packages are defined by the developers as per their application requirement.

To define user defined packages in java, we have to use the following syntax:-

package Package-Name;

To use the above package declaration syntax we have to use the following two conditions.

1) Package declaration statement must be first statement in java files.

Q. Is it possible to provide more than one package declaration statement within a single java file?

Ans:-

No, it is not possible to provide more than one package declaration statement within a single java file; because package declaration statement must be first statement in every java file.

2) Package names must be unique, they must not be sharable and they must not be duplicated.

To provide package names, java has given a convention like to include Our company domain name in reverse of package name.

Ex:- package com.durgsoft.icici.transactions;

com.durgsoft → Company domain name in Reverse
icici → Project/client name

transaction → Module name.

3) Import Section:-

The main purpose of import statement is to make available all the classes and interfaces of a particular package into present java file.

SYNTAX:-

import package-name.*;

→ It able to import all the classes and interfaces from the specified package.

Ex:- import java.io.*;

import package-name.Member-name;

→ It able to import only the specified member from the specified package.

Ex:- import java.io.BufferedReader;

NOTE:- In a single java file, we are able to provide almost one package declaration statement but we are able to provide more than one import statement.]

Q:- To use classes and interfaces of a particular package in java files is it mandatory to import that package.

Ans:- No, it is not mandatory to import package in order to use classes and interfaces of the respective package. We can use a particular package provided classes and interfaces without using import statement, simply by providing classes and interfaces fully qualified names.

NOTE:- Specify class name or interface name along with package name is called as fully qualified name.

Ex:- java.io.BufferedReader
java.util.ArrayList.

java program with import statement:-

17

```
import java.io.*;
BufferedReader br = new BufferedReader(new InputStreamReader(
    System.in));
```

java program without import statement:-

```
java.io.BufferedReader br = new java.io.BufferedReader(new
    java.io.InputStreamReader(System.in));
```

4) Classes/Interfaces Section:-

The main intention to provide class/interfaces is to represent all the real world entities in the form of coding part.

Ex:- Student, Employee, Customer, Account,

In java application we are able to provide any no. of classes and interfaces depending on our application requirement.

5) Main Class Section :-

Main class is a java class contains main() method. The main intention of providing main() method in java application is,

- 1) To define applications logic in java application.
- 2) To define starting point and ending point for the application execution.

18/07/2015

Steps to design First Java application:-

- 1) Install java software
- 2) Select java editor
- 3) Write java program
- 4) Save java file
- 5) Compile java file
- 6) Execute java program.

1) Instal java Software:-

- 1) Download jdk-7-windows-i586.exe from internet.
- 2) Double click on jdk-7-windows-i586.exe file.
- 3) Click on "yes" button.
- 4) Click on "next" button.
- 5) Change JDK installation location from "C:\Program files(x86)\Java\JDK1.7.0" to "C:\Java\Jdk1.7.0" location by clicking on "change" button. (This step is not mandatory, set if there is requirement.)
- 6) Click on "Next" button.
- 7) Change JRE installation location from "C:\Program file(x86)\Java\Jre7" to "C:\Java\Jre7" location by clicking on "change" button.
- 8) Click on "Next" button.
- 9) Click on "Finish" button.

After installation of java software we have to set "path" environment variable in order to make available all JDK tool like javac, java, javadoc, ... to Operating system.

To set "Path" environment variable temporarily we have to

use the following command on Command Prompt.

→ Set Path = C:\Java\JDK1.7.0\bin;

Note:- This is a temporary setup bcz after close the command prompt and reopened, there is a need to set again J.

To set "Path" environment-variable if we use the above approach, then the above setup is available upto the present command prompt, not available to all the Command prompt. To make available "Path" environment variable setup permanently we have to use the following steps.

1) Right click on Computer on desktop.

2) Select Properties.

3) Select Advanced System Settings.

4) Select "Advanced (By default selected)".

5) Select "Environment Variables".

6) Go to user variables block and click on "New" button.

7) Provide the following details.

Variable name: Path

Variable Value: C:\Java\JDK1.7.0\bin;

8) Click on "OK" button.

9) Click on "OK" button.

10) Click on "OK" button.

If we want to switch java from One Version to another version then we have to prepare a separate batch file for each and every version and we have to execute the respective batch file on command prompt.

19/07/2018

Java 6.bat

Set path = C:\Java\Jdk\6.0_11\bin;

Java 7.bat

Set path = C:\Java\Jdk\7.0\bin;

Java 8.bat

Set path = C:\Java\Jdk\8.0\bin;

To execute any batch file, Open command prompt, goto the location where batch files are saved, type batch file name and click on enter button.

D:\Java7am>Java6.bat → Enter button.

2) Select Java Editor:-

Editor is a tool, it will provide very good environment to write java program and to save java programs in our system.

Ex:- Notepad, Notepadplus, Editplus . . .

[NOTE:- In real time application development, Editors are not suggestable, IDEs (Integrated Development environments) are suggestable.

Ex:- Eclipse, MyEclipse, Netbeans, . . .

3) Write java Program :-

To write a java program we have to use some predefined library provided by Java API (Application Programming Interface).

Program:-

```

public class FirstApp {
    public static void main(String[] args) {
        System.out.println("First java Application");
    }
}

```

Diagram illustrating the components of the Java program:

- Access Modifier:** public
- Keyword:** class
- Class Name:** FirstApp
- Return-type:** void
- Method name:** main
- Parameters:** String[] args
- Predefined class name:** System
- Ref. Variable:** out
- Predefined method:** println
- User defined variable:** First java Application
- String data:** "First java Application"

Q) Save java File:-

To save java file we have to use the following rules and regulations.

- 1) If our present java file contains any public class, public abstract class, public interface and public enum then we must save that java file with public element name only. If violates this rules then compiler will rise an error.
- 2) If no public element [class, abstract class, Interface enum] is identified in present java file then it is possible to save java file with any name like abc.java, xyz.java, in this case, instead of saving java file with any other name it is suggestible to save java file with main class name [A class Contains main() method]
- Q) Is it possible to provide more than one public class with in a single java file?

Ans:-

No it is not possible to provide more than one public class with in a single java file, because, if we provide more than one public class with in a single java file then we must save java file with more than one name, it is not possible in all the operating sys. Every java file is able to allow almost ~~one~~ one public element [class, abstract class, interface, enum].

Ex 1: File Name:- abc.java

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

Status:- Valid but not suggestible.

Ex 2: File Name: Test.java

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

Status:- Valid

Ex 3: file Name: Test.java

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

Status:- Invalid

Ex4: file name : A.java

```

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

```

Status:- Invalid.

Ex5: File Name :- Test.java

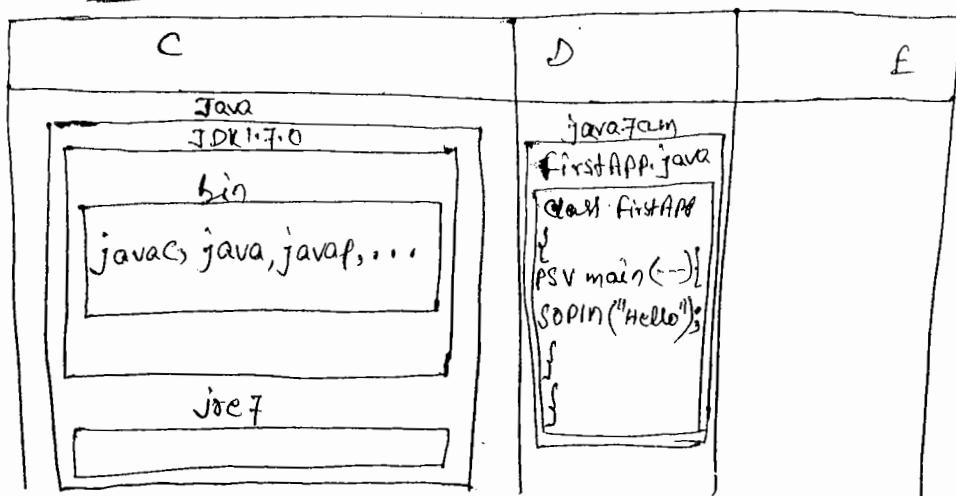
```

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

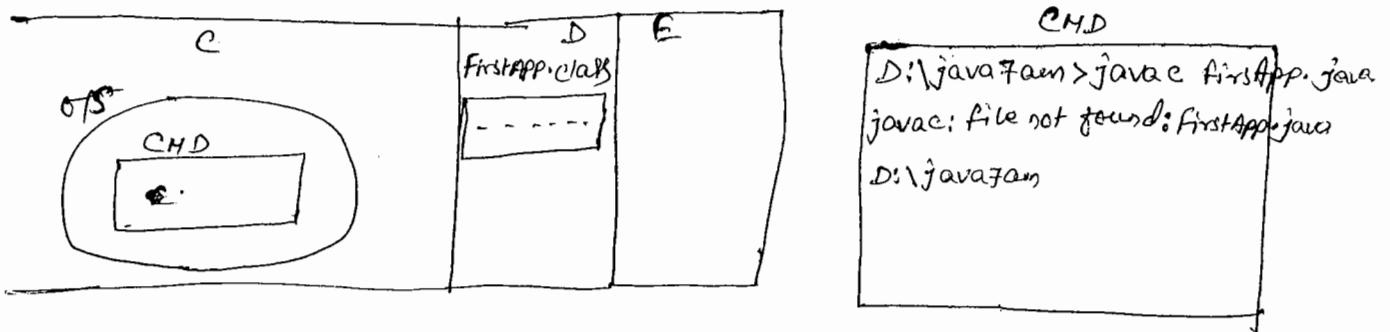
```

Status: Valid.

20/07/2015

5) Compile java file :-

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.



for javac :-

path = ... ; ... ; C:\Java\jdk1.7.0\bin;

Date :- 21/07/2015

The main purpose of compiling java file is,

- 1) To check developer's mistakes in java programs.
- 2) To translate java program from high level representations to [User Understandable representations] to low level representations [JVM understandable representations].

To perform compilation in java applications, we have to use the following command.

javac file-name.java

Ex:-

D:\javafam>javac firstApp.java .

If we provide the above command on command prompt then Operating System will perform the following actions.

- 1) Operating System will take "javac" command from command prompt.
- 2) OS will search for "javac" command in its predefined command list and at the locations referred by "Path" environment variable.

3) If the required "javac" is not identified at all the above locations then OS provide the following message.

"javac is not recognized as an internal or external
operable program or batch file".

(NOTE:- To make available all jdk tools to the OS we have to ~~search~~ set "Path" environment variable to "C:\java\jdk1.7.0\bin".

D:\javatam>set path = C:\java\jdk1.7.0\bin;

4) When OS identified "javac" at the above specified location then OS will execute javac, with this java compiler software will be activated and java compiler will perform the following actions.

a) Compiler will take java file name from command prompt which we specified along with "javac" command.

b) Compiler will search for the required file at current location.

c) If the required file is not identified at the current location then Compiler will provide the following message on command prompt.

javac: file not found: FirstApp.java

d) If the required source file is identified at current location then Compiler will perform compilation right from the starting point of the file to ending point of the file.

e) If any compilation error is identified in compilation then compiler will display error message on command prompt.

f) If no compilation error are identified then compiler will generate .class files.

[Note:- Generating no. of .class files is not at all depending on the no. of java files which we compiled, it is completely depending on the no. of classes, abstract classes, interfaces, enums and inner class which we used in java files.

→ javac -d Target_location File_name.java

Ex:- D:\javafun>javac -d C:\abc FirstApp.java.

If we want to compile java file from current location and if we want to send the generated .class file to a particular target location then we have to use the following command.

If we provide package declaration statement in present java file and if we want to create folder structure w.r.t the package name in order to store .class files then we have to use "-d" option along with "javac" command.

D:\javafun\FirstApp.java

```
Package com.durgsoft.core;
```

```
enum E {
```

```
{
```

```
}
```

```
Interface I
```

```

}
}

abstract class A
{
}

class B
{
}

class C
{
}

}

class FirstApp
{
    public static void main(String[] args)
    {
        System.out.println("First java application");
    }
}

```

D:\javazam>javac -d c:\abc FirstApp.java

If we want to compile all the java files which are available at the current location then we have to use the following command.

D:\javazam>javac *.java

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

→ If we want to compile all the java files which are having a common prefix, then we have to use the following command.

```
D:\javatam>javac Student*.java
```

If we want to compile java files which are having a common postfix then we have to use the following command.

```
D:\javatam>javac *Address.java
```

If we want to compile all java files which are having a common word then we have to use the following command.

```
D:\javatam>javac *Account*.java
```

22-07-2018

6) Execute java Application :-

If we want to execute java programs then we have to use the following command and command prompt ~~is at~~ at the location where main class .class file is available.

- `java Main-Class-Name .`

Ex:-

```
D:\javatam>java firstApp
```

When we use the above command on command prompt then OS will take "java" command from command prompt and OS will execute "java" command by getting location information

from "Path" environment variable.

When OS is executing "java" command automatically, JVM software will be activated and JVM will perform the following actions.

- 1) JVM will take main class name from Command Prompt.
- 2) JVM will search for main class .class file at current location, at java predefined library and at the locations referred by "Classpath" environment Variable.
- 3) If the required main class .class file is not available at all the specified locations then JVM will provide the following.

Java6 : Java.lang.NoClassDefFoundError: FirstApp

Java7 : Error: Could not find or load main class FirstAPP.

[Note] :- If the required main class .class file is available at some other location then to make available that .class file to JVM we have to set "classpath" environment Variable.

' D:\javatutorial>set classpath=E:\abc;

- 4) When JVM identifies main class .class file at either of the above locations then JVM will load main class bytecode to the memory, this phase is called as "Class Loading", this would be performed by "Class Loader" component existed inside JVM.

5) After loading main class bytecode to the memory, JVM will search for main() method, either if main() method is not identified then JVM will provide the following.

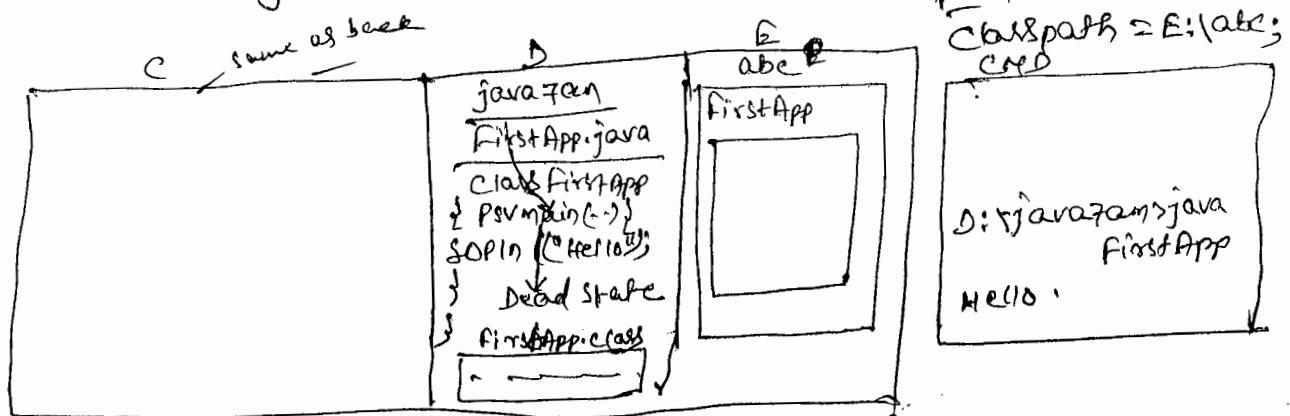
java6: Java.lang.NoSuchMethodError: main

java7: Error: Main method not found in class FirstApp,
please define the main method as:
public static void main (String[] args).

6) If main() method is identified in main class bytecode then JVM will create a thread to access main() method called as "Main Thread".

7) JVM will execute main() method by passing "Main Thread" and generate some output as per the implementation of main() method.

8) When main thread reached to main() method ending point then main thread will get dead state, with this, JVM will stop all of its external processes and JVM will go to shutdown mode.



Language Fundamentals:-

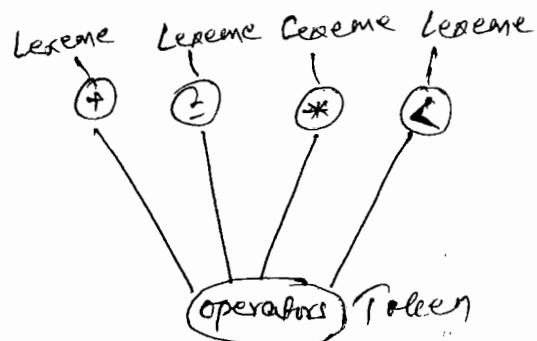
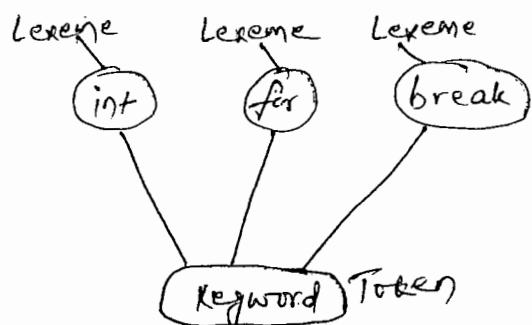
- 1) Tokens
- 2) Data types
- 3) Type Casting
- 4) Java Statement
- 5) Arrays

To design java applications we need some building blocks from java programming language. To design java applications java has provided the required building blocks in the form of the following list.

1) Tokens:

The smallest logical unit in java programming is called as "lexeme".

The collection of lexemes come under a particular group is called as "Tokens".



Q) Find the no. no. of lexemes and no. of tokens in the following expression
 int a=b+c*d;

Ans.

int, a, =, b, +, c, *, d, ; → 9 lexemes.

int → Data type.

a, b, c, d → Identifiers

=, +, * → Operators.

; → Special symbol/Terminator.

Total → 4 types of tokens.

To design java applications, java has provided the following Tokens.

① Identifiers

② Literals

③ Keyword / reserved words.

④ Operators.

2) Identifiers:-

Identifiers is a name assigned to the programming elements like Variables, methods, classes, ...

To use identifiers in java applications, java has provided the following rules and regulations.

① Identifiers must not start with a number.

Identifiers may start with an alphabet, -Symbol and \$ symbol, but the subsequent symbol may start with an alphabet, a number, -Symbol and \$ symbol.

Ex:-

int .eno = 111; → Valid
 int geno = 999; → Invalid
 String -ename = "Durga"; → Valid
~~float~~ pesal = 50000.0f; → Valid
 String emp-Name = "Durga"; → Valid
 float emp\$sal = 50000.0f; → Invalid
 int emp@-9 No = 999; → Valid

2) Identifiers are not allowing all the Operators and all the Special symbols except - symbol and \$ symbol.

Ex:-

int emp-NO=111; → Invalid
 String emp-Id = "E-111"; → Invalid
 float emp+sal = 50000.0f; → Invalid
 String emp·Addr = "Hyd"; → Invalid
 String emp@Hyd = "Durga"; → Invalid.
 String emp#Email = "durga@soft.com"; → Invalid
 float esal\$= 50000.0f; → Valid

3) Identifiers are not allowing spaces in the middle.

Ex:-

for Name(-); → Invalid
 forName(); → Valid
 get Input- Stream(); → Invalid
 getInputStream(); → Valid

4) Identifiers should not be duplicated with in the same scope, but Identifiers may be duplicated in two different scopes.

Ex:-

```
class A
{
    float f = 22.22f;
    double f = 24.284; → Invalid
    long l = 40; Valid
    void m1()
    {
        int i = 10;
        double f = 234.456; → valid.
        long l = 40; → Invalid.
    }
}
```

5) In java applications, it is possible to use all predefined class names and abstract class names and interface names as identifiers.

Ex:- `int Exception = 10;` ① Class Test
`System.out.println(Exception);` `public static void main(String[] args)`
 { } States:- No compilation error
 OP:- 10

② Class Test

```
{ public static void main (String[] args)
{
    String String = "String";
    SOP (String);
}
States:- No compilation error
OP:- String.
```

③ Class Test

```

    {
        public static void main (String [] args)
        {
            int system = 10;
            System.out.println (System);
        }
    }

```

Output:- Compilation Error.

Reason:-

In java applications, if we declare any predefined class name or interface name as an identifier or variable then we must use that class name or interface name as variable only, not as original class name, if we use that as original class name then compiler will rise an error.

In the above context, if we want to use the respective class name as original class name then we have to use fully qualified name of the class.

Note:- Specifying class name or interface name along with package name is called as fully qualified name.]

Ex:- `java.io.BufferedReader`
`java.util.ArrayList`

Ex:-

```

    class Test
    {
        public static void main (String [] args)
        {
            int system = 10;
            java.lang.System.out.println (System);
            System.out.println (System + 10);
            java.lang.System.out.println (System);
        }
    }

```

Suggestion:-

① Along with the above rules and regulations, java has provided the suggestions to declare identifiers.

1) Identifiers must be meaningful, they must reflect a particular meaning.

Ex:-

String xxx = "abc123" → Not suggestible.

String accNo = "abc123" → Suggestible

2) There is no length restriction in identifiers, we can declare identifiers with any length but it is suggestible to manage identifiers length around 10 symbols.

Ex:-

String temporaryemployeeaddress = "Hyd" → Not suggestible

String tempEmpAddr = "Hyd" → Suggestible

3) If we have multiple words within a single identifier then it is suggestible to separate multiple words with different rotations like '-' symbols.

Ex:-

String tecnEmpAddr = "Hyd"

String temp-Emp-Add = "Hyd"



2) Literals:-

Literal is a constant assigned to the Variables.

Ex:-

int a=10;
 int → data type
 a → Variable
 = → Operator
 10 → Constant (Literal)
 ; → Special symbol/Terminator.

To design java applications, java has provided following literals.

1) Integral/Integer Literals:-

byte, short, int, long → 10, 20, 35, ...
 char → ~~23, 34, 45, 49, 'C', 'a'~~

2) Floating Point literals:-

float → 12.22f, 34.95f, ...
 double → 234.34, 456.44, ...

3) Boolean literals:-

boolean → true, false

4) String literals:-

String → "abc", "xyz", "def", ...

~~double d~~

To improve readability while reading numbers, JAVA7 version has given a flexibility like to include '-' symbols in the numbers.

Ex:- double d = 12.34-56-789.23456;

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

If we compile the above instruction, compiler will remove '-' symbols from number, compiler will make that no. as Original no., compiler and JVM will process that number as Original no.

Number Systems in java:-

To represent numbers in any programming language we have to use the following number systems.

- 1) Binary Number System
- 2) Decimal
- 3) Octal
- 4) Hexadecimal Number System.

java is able to support all the above no. systems but the default no. system in java is "Decimal no. system".

1) Binary Number System:-

If we want to represent binary no. system in java applications then we have to prepare number by using the symbols like 0's and 1's and the respective no. must be prefixed with either '0b' or '0s'.

Ex:-

int a = 10; → Valid but it is not binary no.
int b = 0b10; → Valid
int c = 0B1010; → Valid
int d = 0b1020; → Invalid.

Octal Number System :-

If we want to represent any number in Octal number system then we have to prepare that no. by Using the symbols like 0, 1, 2, 3, 4, 5, 6 and 7 but the number must be prefixed with 0 (zero).

Ex:-

int a=10; → Valid but it is not Octal numbers

int b=012345; → Valid

int c=034567; → Invalid

int d=0678 → Invalid.

Hexa decimal Number System :-

If we want to represent any no. in Hexa decimal no. system then we have to prepare that number with the symbols like 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, and f but the no. must be prefixed with 0x or 0X.

Ex:-

int a=10; → Valid but it is not Hexa decimal

int b=0x123456; → Valid

int c=0x56789; → Valid

int d=0xabcde; → Invalid

int e=0xefg; → Invalid.

If we represent any no. in the above no. sys, then Computer will convert that no. from the respective no. system into decimal no. system, where Computer and JVM will process that number as decimal number.

26/07/2015

3) Keywords / Reserved Words:-

Keyword is a predefined word, it will have both word recognition and internal functionality.

Reserved word is a predefined word, it will have only word recognition with out internal functionality.

Ex:- goto, const.

To design java application java has provided the following list of keywords.

1) Data type and return types:-

byte, short, int, long, float, double, char, boolean, void, ...

2) Access modifiers:-

public, protected, private, static, final, abstract, native, volatile, transient, synchronized, StrictFP; ...

3) Flow Controllers :-

if, else, switch, case, default, for, while, do, break, continue, return, ...

4) Class/ Object Related:-

class, extends, interface, implements, enum, new, this, super, package, import, ...

5) Exception handling:-

throw, throws, try, catch, finally.

==

4) Operator's.

4) Operators:-

Operator is a symbol, it will perform a particular operation over the provided Operands.

To design java applications, java has provided the following Operators:-

1) Arithmetic Operators:-

+, -, *, /, %, ++, --

2) Assignment Operators:-

=, +=; -=, *=, /=, %=

3) Comparison Operators:-

==, !=, <, >, <=, >=

4) Boolean Logical Operators:-

&, |, ^

5) Bitwise Logical Operators:-

&&, ||, ^, <<, >>

6) Short Circuit Operators:-

&&, ||

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

7) Ternary Operator:-

expr1? expr2: expr;

Ex:-

Class Test

{ public static void main(String[] args)

int a = 5; ~~op: 5~~

SOP(a); op: 5

SOP(a++); op: 5

SOP(++a); op: 7

SOP(a--); op: 7

SOP(--a); op: 5

SOP(a); op: 5

a [5 6 7 8 5]

7

}

Q

Ex:-
Class Test

{ public static void main(String[] args).

int a = 7;

SOP(a++ - ++a);

}

Ans = -2.

Ques

int a = 7; a [7 8 9]

a++ - ++a

7 - 9

a = -2

27-07-2015

Ex:- if $a = 8;$

S.O.P ($++a + a-- * --a - + a$);

~~$\cancel{+} \cancel{-}$~~

$9 + 9 * 7 - 8$

= 64

88888

Ex:-

if $a = 5;$

S.O.P ($++a - a-- + (a-- + a--) * (++a - a--) + (++a - a--)$);
 $(8 - 8 + 5 - 4) * (8 - 8 + 4 - 4)$

= 0

78886

Example of Boolean Operator :-

A	B	$A \otimes B$	$A \mid B$	$A \wedge B$
T	T	T	T	F
T	F	F	T	T
F	T	F	T	T
F	F	F	F	F

Ex:-

Boolean b1 = true;

Boolean b2 = false;

S.O.P ($b1 \otimes b1$); true

S.O.P ($b1 \otimes b2$); false

S.O.P ($b2 \otimes b1$); false

S.O.P ($b2 \otimes b2$); false

S.O.P ($b1 \mid b1$); true

S.O.P ($b1 \mid b2$); true

S.O.P ($b2 \mid b1$); true

S.O.P ($b2 \mid b2$); false

S.O.P ($b1 \wedge b1$); false

S.O.P ($b1 \wedge b2$); true

S.O.P ($b2 \wedge b1$); true

S.O.P ($b2 \wedge b2$); false.

}

}

Ex:- Bitwise Operators:-

int a = 10; → 1010 (binary conversion)
int b = 2; → 0010 (binary conversion)

$$S.O.P (a/b); \rightarrow 0020 \rightarrow 2$$

$$S.O.P \left(a|b \right); \rightarrow 1010 \rightarrow 10$$

$$S \cdot O \cdot P (Q^k b); \rightarrow 2000 \rightarrow 8$$

S.O.P (acc \leq b): —

S-0-1P (azzb); 00001010
S-0-1P (azzb); 00101000

S.O.P. ($a \gg b$); $\boxed{00101000} \rightarrow \boxed{0}$

Q

oo
~~147~~

00001010

00000010 → ~~1~~2

Boolean Truth Table :-

A	B	$A \& B$	$A \mid B$
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F

Ex:- for I and II

$$\text{int } Q = 10; \boxed{2011}$$

$$\text{int } b = 10; \boxed{1011}$$

\Rightarrow if $((a + \frac{t}{T}) = 10) \text{ or } ((b + \frac{t}{T}) = 10)$
 $8.0 \cdot P/ln(\frac{a + " " + 5}{11} \frac{10}{11})$

b) $\neg ((a \stackrel{10}{+} \stackrel{10}{=} 10) \vee (\underline{b \stackrel{10}{+} \stackrel{10}{=} 10}))$

S.O.P17 (a+) " + b);
{} 11 (10)

↳ If VM is conforming
if there is unnecessary
execution is performed
then it can't perform
only ~~if~~ necessary

int a=0; 18 //

int a = 10; 10 11

$$S.O.PIn \left(a_{21}^{+} \right) \left(+b_{21} \right)$$

b) $\{f((a++! \geq 10) \& (b++! \geq 10))\}$

$$S.O.P(n) \left(\begin{array}{cc} a & " + b \\ 1 & 0 \end{array} \right),$$

A	B	$A \oplus B$	$A \mid B$
T	T	F	T
T	F	T	T
F	T	T	F
F	F	F	F

Short Circuit Operators:

The main purpose of Short-Circuit Operator is to improve performance of the java applications.

There are two short-circuit operators.

1) && 2) ||

Difference:-
| vs || :-

In the case of Boolean 'OR' Operator, if the first operand value is true then it is not required to check second operand value, we can predict overall expression result is 'true'. In the case of 'I' operator, even though first operand value 'true' still JVM will execute second operand value then only JVM will predict overall expression result, in this case, executing second operand is unnecessary, it will reduce performance of the java applications.

In the case of '||' operator, if the first operand result is 'true' then JVM will not execute second operand and JVM will decide the result of overall expression is 'true'. This approach will improve java application performance.

[Note:- Some shortcut operation is going on in between & and && operators but the first operand result must be false.]

[Note:- If the first operand result is not sufficient to predict overall expression result then JVM must execute second operand value in the case of I, ||, & and &&.]

Data Types:-

java is strictly a typed programming language, where in java applications, we have to decide which type of data we are going to represent before representing data. In this context, to specify the type of data which we are representing we have to use "data types".

In java application, data types are able to provide the following advantages.

- 1) We are able to identify memory sizes to store data on the basis of data types.
- 2) We are able to identify range values which we are going to assign to the variables on the basis of data types.

→ To design java application, java has provided the following data types:-

- 1) Primitive data types / Primary datatypes.

a) Numeric data types

→ Integral / Integer data types.

byte → 1 byte → 0

Short → 2 bytes → 0

int → 4 bytes → 0

long → 8 bytes → 0

2) Non-Integral Data types:-

float \rightarrow 4 bytes \rightarrow 0.0f
 double \rightarrow 8 bytes \rightarrow 0.0

3) Non-Numerical data types:-

Char \rightarrow 2 bytes \rightarrow ' ' [single space]
 boolean \rightarrow 1 bit \rightarrow false.

User defined data types/secondary data types :-

All arrays, All class types, All interface types, ...
 The default value for user defined data type is "null".

To calculate range values on the basis of the data type, java has given the following formula:

$$-2^{n-1} \text{ to } 2^{n-1} - 1$$

Where "n" is no. of bits

Size:-

$$\text{size:- 1 byte = 8 bits}$$

$$-2^{8-1} \text{ to } 2^{8-1} - 1$$

$$-2^7 \text{ to } 2^7 - 1$$

$$-128 \text{ to } 128 - 1$$

$$-128 \text{ to } 127$$

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

To represent minimum and maximum values for each and every data type, java has provided the following constant variable at all the wrapper classes.

MIN-VALUE & MAX-VALUE.

Note:- Classes representation of all the primitive data types are called as wrapper classes.

<u>Primitive</u>	<u>Wrapper classes</u>
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character

Ex:- class Test

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

Type Casting:-

The process of converting data from one data type to another data type is called as Type Casting.

There are two types of type casting in java:-

1) Primitive Data types type Casting.

2) User defined data types type casting.

The process of converting data from one user defined data type to another user defined data type is called as User defined data types type Casting.

To perform user defined data types type casting we have to provide either "extends" relation or "implements" relation between two user defined data types.

1) Primitive data types type Casting:-

The process of converting the data from one primitive data type to another primitive data type is called as primitive data types type casting.

There are two types of primitive data types type casting.

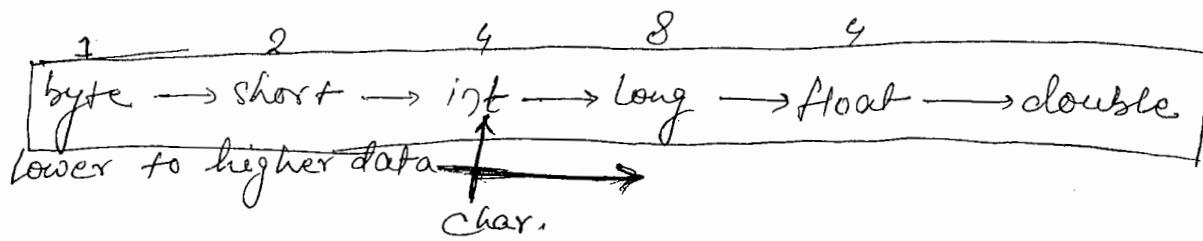
1) Implicit Type Casting

2) Explicit Type Casting.

1) Implicit Type Casting:-

The process of converting the data from lower data type to higher data is called as Implicit Type Casting.

To cover all the possibilities of implicit type casting java has provided the following chart:-



Ex:- 1)

```
byte b=10;
int i=6;
System.out.println(b+" "+i);
```

Status:- No Compilation Error.

OP:- 10 10

If we compile the above code then compiler will check whether right side variable data is compatible with left side variable data type or not, if not, compiler will rise an error like "Possible loss of precision".

If right side variable data type is compatible with left side variable data type then compiler will not rise any error and compiler will not perform any type casting:

Note:- In java, all lower data types are compatible with higher data types and all higher data types are not compatible with lower data types.

When we execute the above code, jvm will perform the following two actions-

1) Type Casting:- Converting right side Variable data to to Left side Variable data type implicitly.

2) Value Copy:-

Transferring value from right side Variable to Left side Variable.

³⁴
Note:- Type checking is the responsibility of compiler and type casting is the responsibility of programmer.

Note:- To perform implicit type casting, simply we have to assign lower data type variable to higher data type variable.

Ex:-
a) int i=10;
 byte b=e;
 S.O.P(i+" "+b);

Status:- Possible loss of precision.

b) byte b=65;
 char c=b;
 S.O.P(b+" "+c);

Status:- Compilation Error, possible loss of precision.

c) char c='A';
 short s=c;
 S.O.P(c+" "+s);

Status:- Compilation Error, possible loss of precision.

Reason:-

In implicit type casting, conversions between byte and char, short and char are not interested.

d) char c='A';
 int i=c;
 S.O.P(c+" "+i);

Status:- No compilation Error.

Op:- A 65

30-07-2015

e) long l=10;
float f=1;
S.O.PLN(l+" "+f);

Status:- No compilation error

OP :- 10 10.0

(f) byte b=128;
S.O.PLN(b);

Status:- Compilation Error, possible loss of precision.

Reason:-

If we assign any value to byte variable or short variable which is greater than the max values of the respective data types then the provided value is treated as an integer value then compiler will rise an error.

g) byte b1=60;
byte b2=70;
byte b=b1+b2;
S.O.PLN(b);

Status:- Compilation error, possible loss of precision.

h) byte b1=10;
byte b2=20;
byte b=b1+b2;
S.O.PLN(b);

Status:- compilation error, possible loss of precision.

Reason:- If x, y and z are three primitive data types.
 $x+y=z$

- 1) If x and y belongs to {byte, short, int} then z should be "int".
- 2) If either x or y or both belongs to {long, float, double} then z should be higher(x, y)

Ex:-

$$\begin{aligned} \text{byte + byte} &= \text{int} \\ \text{byte + short} &= \text{int} \\ \text{byte + int} &= \text{int} \\ \text{short + long} &= \text{long} \\ \text{int + long} &= \text{long} \\ \text{long + float} &= \text{float} \\ \text{float + double} &= \text{double} \end{aligned}$$

2) Explicit type Casting:-

The process of converting data from higher data type to lower data type is called as explicit type casting.

To perform explicit type casting we have to use the following pattern:

$P\ a = (Q)\ s;$

where 's' variable data type should be higher than 'P':

where Q must be either same as 'P' or lower than 'P':

Ex:- $\text{int } i=10;$
 $\text{byte } b=(\text{byte})i;$
 $\text{s.o.p}(i"\&s);$

Status → xdo compilation error

Output: 10 10

b) `int i=10;`

`short s=(byte)i;`

`S.O.P1N(i+" "+s);`

Status:- No compilation error

Output 10 10.

c) `byte b=65;`

`char c=(char)b;`

`S.O.P1N(b+" "+c);`

Status:- No compilation error.

Off :- 65 A

d) `Char c='A'`

`short s=(byte)c;`

`S.O.P1N(c+" "+s);`

Status:- No compilation error.

Output :- A 65

Reason:-

In `eg` implicit type casting, conversions are existed between `char` and `byte`, `char` and `short` but in explicit type casting conversions are existed between `char` and `byte`, `char` and `short` by force.

eg:- `int i=130;`

`byte b=(byte)i;`

Status:- No compilation error

Off :- ~~130~~ -126.

Value	Off
128	-128
129	-127
130	-126

f) `byte b1=10;`
`byte b2=10;`
`byte b=(byte)(b1+b2);`
`S.O.PN(b);`

Status: No compilation error.
OP :- 20

g) `float f=12.22f;`
`long l=10;`
`long l1=l+f;`
`S.O.PN(l1);`

Status- Compilation error, possible loss of precision.

h) `float f=12.22f;`
`long l=10;`
`long l1=l+(long)f;`
`S.O.PN(l1);`

Status: No compilation error.
OP :- 22

i) `double d=22.22;`
`byte b=(byte)(short)(int)(long)(float)d;`
`S.O.PN(b);`

Status- No compilation error.

OP :- 22

3

03/08/2015

Java Statements:-

Statement is a collection of Expressions.

To design java applications, java has provided the following list of statements.

1) General Purpose Statements:-

Declaring Variables, methods, classes, ...

Accessing Variables, methods, ...

Creating Objects, ...

2) Conditional Statements:-

if, switch

3) Iterative Statements:-

for, while, do-while.

4) Transfer Statements:-

break, continue, (Return)^{→ remain}

5) Exception Handling:-

throw, try - catch - finally.

Conditional Statements:-

These java statements are able to allow to execute a block of instructions under a particular condition.

Ex:- if, Switch.

① "if" Conditional Statement:-

37

SYNTAX:-

1) if (condition)
 {
 — instructions —
 }

2) if (condition)
 {
 — instructions —
 }
 else
 {
 — instructions —
 }

3) if (condition)
 {
 — instructions —
 }
 else if (condition)
 {
 — instructions —
 }
 —
 —
 —
 else
 {
 — instructions —
 }

Note:- If we want to provide single statement as body to if and else then curly braces ({}) are optional.

Ex:-1
Class Test

```
{  
    public static void main (String [] args)  
    {  
        int i=10;  
        int j;  
        if (i==10)  
        {  
            j=20;  
        }  
        System.out.println(j);  
    }  
}
```

Status :- Compilation error,

```
2) int i=10;  
    int j;  
    if (i==10)  
    {  
        j=20;  
    }  
    else  
    {  
        j=30;  
    }  
    System.out.println(j);  
}  
  
Status :- No compilation error  
Output :- 20.
```

```

3) int i=0;
int j;
if (i==10) Hi.
{
    j=20;
}
else if (i==20) Hi.
{
    j=30;
}
System.out.println(j);
}

```

States:- Compilation error.

```

4) int i=10;
int j;
if (i==10) // 50%.
{
    j=20;
}
else if (i==20) // 25%
{
    j=30;
}
else // 25%.
{
    j=40;
}
System.out.println(j);
}

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

States: No Compilation error
OP:- 20.

```

5> int j;
   if (true)
   {
       j=20;
   }
   System.out(j);
}

```

Status:- No compilation error

O/P :- 20

```

6> final int i=10;
   int j;
   if (i==10)
   {
       j=20;
   }
   System.out(j);
}

```

Status:- No compilation error

O/P :- 20.

Reasons:-

- 1) In java, ~~all~~ only class level variables are having default values, local variables are not having default values.
- 2) If we declare any local variable then we must provide initialization for that local variable either at the same declarative statement or atleast before accessing that local variable.

Ex:-

```

class A{
    int i=10;
    int j;
    void m1()
    {
        int k=30;
        int l;
        int m;
        System.out(k); → NO CB, O/P:30
        l=40
        System.out(l) → NO CB, O/P:40
    }
}

```

`S.O.P(m);` → CE, No value for 'm'.

`S.O.P(n(l));` → NO CE, OP: 10

`S.O.P(n(j));` → NO CE, OP: 0.

{
}

3) In java there are two types of conditional expressions.

1) Constant expressions.

2) Variable expressions.

1) Constant Expressions:-

These expressions must include Only Constant Values (including final variables also).

Eg:- 1) `if (10 == 10)`

2) `final int i=10;`
`if (i==10)`

3) `if (true)`

4) `int i=10;`

`if (i==10)` → Variable expressions.

[Note:- In java applications, constant expressions are evaluated by the compiler, not by JVM.

2) Variable Expressions:-

These expression must have atleast One Variable.

Eg:- 1) `int i=10;`
`if (i==10)`

2) `int i=10;`
`int j=10;`
`if (i==j)`

04-08-2015

3) if ($10 == 10$) \rightarrow constant expression.

(Note:- Variable expressions are executed by JVM, not by compilers.)

2) Switch:- ("switch" conditional Statement):-

In java applications, 'if' conditional statement is able to allow single condition checking but 'switch' conditional Statement is able to allow multiple conditions checking.

→ In general, Switch is utilized in menu driven applications.

Syntax:-

Switch (Var_Name)

{

Case 1:

— Statements —

break;

Case 2:

— Statements —

break;

—

Case n:-

— Statements —

break;

default:

— Statements —

break;

}

Ex:-

```

Class Test
{
    public static void main(String[] args)
    {
        int i=10;
        switch(i)
        {
            case 5:
                System.out.println("Five");
                break;
            case 10:
                System.out.println("Ten");
                break;
            case 15:
                System.out.println("Fifteen");
                break;
            case 20:
                System.out.println("Twenty");
                break;
            default:
                System.out.println("Default");
                break;
        }
    }
}

```

\Rightarrow O/P:- ~~Five~~ 10

Rules to Prepare Switch in java programming:-

1) Switch is able to allow the data types byte, short, int and char as parameter, Switch is unable to allow the data types like long, float and double.

key:-

byte b = 10;

switch (b)

{

— Case and default —

}

Status:- Valid

2) int i=10;

switch (i)

{

— Case and defaults —

}

Status:- Valid.

3) long l=10;

switch (l)

{

— Case and Defaults —

}

Status: Compilation error, possible loss of precision.

```
4) char c = 'A';
   switch (c)
   {
       case and default —
   }
```

Status: No compilation error.

Note:- Upto java6 version it is not possible to provide String data type as parameter to Switch, but, JAVA7 is able to allow String data type as parameter to Switch.

Class Test

```

public static void main (String [ ] args)
{
    String str = "AAA";
    switch (str)
    {
        case "AAA":
            System.out.println ("AAA");
            break;
        case "BBB":
            System.out.println ("BBB");
            break;
        case "CCC":
            System.out.println ("CCC");
            break;
        case "DDD":
            System.out.println ("DDD");
            break;
        default:
            System.out.println ("Default");
    }
}
```

→ O/P :- "AAA"

2) In switch all cases are optional, default case is optional. It is possible to provide switch without cases and with default, it is possible to provide with out default and with cases, it possible to provide switch without cases and

Ex:- 1)

```
int i=5;
switch (i)
{
    case 5:
        S.O.Pln("Five");
        break;
}
```

Status:- No Compilation error
OP:- Five.

2)

```
int i=5;
switch (i)
{
    case 5:
        S.O.Pln("Five");
        break;
}
```

Status:- No Compilation error
OP:- Five.

3)

```
int i=25;
switch (i)
{
    case 5:
        S.O.Pln("Five");
        break;
}
```

Status:- No Compilation error
OP:- No Output.

```
4) int i=10;
switch(i);
```

```
{  
}
```

Status: No compilation Error, but no use.

- ③ In switch break statements is optional, in this case, JVM will execute all the cases right from matched case to end of the switch if we are not providing break statement.

Class Test

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

```
{  
int i=10;  
switch(i)
```

Case 5:-

```
System.out("Five");  
break;
```

Case 10:-

```
System.out("Ten");  
break;
```

Case 15:-

```
System.out("Fifteen");  
break;
```

Case 20:-

```
System.out("Twenty");  
break;  
default:
```

```
System.out("Default");
```

{ m }

O/P:- Ten, Twenty, Fifteen,
Twenty and Default.

Q) In switch we have to provide all the case values with in the range of the data type which we passed as parameter to switch.

Ex:- → Class Test

~~W~~ Public static void main (String [] args)

{

byte b = 125;

switch (b)

{

case 125:

S. O. P1n ("125");

break;

case 126:

S. O. P1n ("126");

break;

case 127:

S. O. P1n ("127");

break;

case 128:

S. O. P1n ("128");

break;

default:

S. O. P1n ("Default");

break;

}

}

[NOTE:- 128 is not possible because
default range of byte is -128 to
127].

5) In switch, all the case values must be constants.
(Final constants are also possible), case values must not be variables.

⇒ class Test
{ public static void main(String args)
{
 final int i=5, j=10; k=15, l=20
 Switch (i0)
 {
 Case i:
 System.out.println ("Five");
 break;
 Case j:
 S.O.Pln ("Ten");
 break;
 Case k:
 S.O.Pln ("Fifteen");
 break;
 Case l:
 S.O.Pln ("Twenty");
 break;
 default:
 S.O.Pln ("Default");
 break;
 }
}

Status:- No Compilation Error

O/P:- 10

Iterative Statements:-

These statements are able to allow to execute a set of
1) for instructions repeatedly on the basis of a Condition.
2) while These are three types of iterative statements.
3) do-while.

1) for :-

Syntaxes:-

```
for(expr1; expr2; expr3)
{
    — Statements —
}
```

[Note:- If we want to provide single statement in for loop body then curly braces are optional.]

Ex:- 1>

```
for (int i=0; i<10; i++)
{
    S.O.P(i)
}
```

Result:-

expr1 → 1 time
expr2 → 11 times
expr3 → 10 times
body → 10 times

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

2) `int i=0;
for(; i<10; i++)
{
 S.O.Println(i);
}`

3) `int i=0;
for(S.O.Println("Hello"); i<10; i++)
{
 S.O.Println(i);
}`

Status:- No Compilation error
OP:- 0 1 2 3 4 5 6 7 8 9

Status:- No Compilation Error
OP:- Hello 0 1 2 3 4 5 6 7 8 9

Reason:- In for loop syntax, expr1 is optional, it is possible to write for loop without expr1, we can write any statement as expr1 but always it's suggestible to provide for loop variable increments or decrement kind of statements.

4) `for(int i=0, float f=0.0f; i<10 & f<10.0f; i++, f++)
{
 System.out.println(i + " " + f);
}`

Status:- Compilation Error.

5) `for(int i=0, int j=0; i<10 & j<10; i++, j++)
{
 System.out.println(i + " " + f);
}`

Status:- Compilation Error

⑥

```
for( int i=0, j=0; i<10 & j<10; i++, j++ )  
    S.O.Pn ( i+ " " + j );
```

Output:- No Compilation Error.

O/P :- 0 0
 1 1
 2 2
 3 3
 : :
 9 9

Reason:-

In for loop syntax, expr is able to allow almost one declarative statement, expr will not allow more than one declarative statement. In expr, it's possible to declare more than one loop variable with in a single declarative statement.

⑦

```
for( int i=0; ; i++ )  
    S.O.Pn ( i );
```

Output:- No compilation error

O/P :- Infinite

Ctrl+C → for exit from Cnsl.

8)

```
for (int i=0; S.O.PIN ("Hello"); i++)
{
    S.O.PIN (i);
}
```

Status:- Compilation error

required :- Boolean

found :- void

Reason:-

In for loop expr2 is optional, we can write for loop even without expr2. If we write for loop without expr2 then for loop will take "true" value by default even though the default value for boolean variable is false. If we want to write any statement as expr2 then that statement must be boolean statement, it must return either true or false value.

9)

```
S.O.PIN ("Before loop");
```

```
for (int i=0; i>=0; i++)
```

```
{
    S.O.PIN ("Inside loop");
}
```

```
S.O.PIN ("After loop");
```

Status:- No compilation error

O/P:- Before loop
Inside loop (infinite times).

10)

```
S.O.Pln ("Before loop");
for (int i=0; true; i++)
{
    S.O.Pln ("Inside loop");
}
S.O.Pln ("After loop");
```

Status:- CE, Unreachable Statement.

11)

```
S.O.Pln ("Before loop");
for (int i=0; ; i++)
{
    S.O.Pln ("Inside loop");
}
S.O.Pln ("After loop");
```

Status:- CE, Unreachable Statement

Reason:-

If we provide any statement immediately after infinite loop then that statement is called as Unreachable Statement.

Rising Unreachable Statement Error or not is completely depending on how much compiler is able to recognize our loop is infinite loop. If compiler our loop is ~~suspect~~ infinite loop and if compiler identifying every instruction

06-08-2015

immediately after infinite loop then compiler will rise an error like "Unreachable Statement".

If compiler is not recognized our loop is infinite loop then compiler will not rise any error.

Note:- Deciding a loop is an infinite loop or not by the compiler is completely depending on the conditional expression which we provided. If the conditional expression is constant expression and its result is true by the provided loop is infinite loop recognized by the compiler. If the conditional expression is variable loop is an infinite loop.

~~Ex:- Ex:- 12) for(int i=0; i<10) { S.O.P1n (i); i = i+1; }~~

Status:- No CE

OP:- 0 1 2 3 4 5 6 7 8 9

~~13) for(int i=0; i<10; S.O.P1n ("Hello")) { S.O.P1n (i); i = i+1; }~~

Status:- No CE

OP:- 0 Hello 1 Hello 2 Hello 3 Hello 9 Hello

Reason:-

In for loop, expr3 is optional, we can write for loop without expr3, we can use any statement as expr3 but always it is suggestible to provide loop variables increments or decrements operation as expr3.

14)

```
for(;;)
```

}

{ Status:- No CE

OP:- Infinite loop (because information is not provided)

15)

```
for(;;)
```

}

{ Status:- CE

16)

```
for(;;);
```

}

{ Status:- No CE

OP:- infinite loop

17)

```
for(;;)
```

;

{ Status:- No CE

OP:- infinite Loop.

Reason:-

If we don't want to provide statement in for loop body then we must provide either curly braces " {} " if we have not provided both then compiler will raise an error.

In general, in java applications, we are able to use ~~use for~~ for loop when we aware the no. of iterations of any loop before writing loop.

Ex:- To retrieve elements from an array we have to use for loop, because no. of iterations in for loop are equals to array size that we are able to identify before writing loop.

Ex:-

```
int [] a = {1, 2, 3, 4, 5};  
for (int i=0; i<a.length; i++)  
    System.out.println(a[i]);  
}.
```

To retrieve elements from an array or from Collection objects if we use 'for' loops like above then we are able to get the following problems:-

- 1) We have to manage a separate loop variable.
- 2) At each and every iteration conditional ~~Statement~~ expression must be executed, if it's strengthfull expression, it will take more execution time.
- 3) At each and every ~~iteration~~ iteration, increment or decrement Operations over loop Variables must be performed.

q) We have to retrieve array elements on the basis of array index values that is loop variable value, if loop variable value is not proper then there may be a chance to get an exception like "ArrayIndexOutOfBoundsException".

→ All the above drawbacks are able to reduce java application performing while retrieving elements from arrays or from collection objects.

To Overcome all the above problems, JDK5.0, Version has provided an enhancement on "for" loop called as "Each-for" loop.

Syntax:-

```
for(Array-data-Type Var-name; Array-Ref-var)
{
    instructions
}
```

Ex:-

```
int []a = {1, 2, 3, 4, 5}
for (int x: a)
{
    S.O.P(x);
}
```

When JVM encounters the above instruction, JVM will take value by value from the specified array

and assign that values to the specified variable at each and every iteration.

Note:- "for-each" loop is useful for only retrieving elements from an array or from Collection Objects, it's not for normal iterations.J.

=

While loop:-

In java applications when we are not aware the no. of iterations before writing the loop then we have to use "while" loop.

Syntax

```
while (condition)
{
    _____ instructions _____
}
```

Ex:- 1)

```
i=0;
while (i<10)
{
    S.O.P(i);
    i=i+1;
}
```

Status:- NO CE

O/P :- 0 1 2 3 4 5 6 7 8 9

2)

```
int i=0;
while()
{
    S.O.P(i);
    i=i+1
}
```

Status:- CE, illegal start of Expr - section.
O/P:-

Reason:-

In while loop condition expression is mandatory.

3) `final int i=0;`

`while(i<10)`

{
 `S.O.Pn(i++);`

} } Status:- CE

Reason:-

In the case of loops, we must not declare loop variables as final variables, because we must perform increment or decrement operation over loop variables.

4) `int i=0`

`while(i<10)`

{
 `S.O.Pn(i++);`

} } Status:- NO CE

OP :- 0 1 2 3 4 5 6 7 8 9

5) {

`int i=0;`

`S.O.Pn ("Before loop");`

`while(true)`

{
 `S.O.Pn ("Inside loop");`
 `i = i+1;`

} } `S.O.Pn ("After loop");`

} } Status:- CE, Unreachable Statement

do-while:-

1) In case of while-loop there is no guarantee whether loop body is executed minimum one time or not.

In case of do-while loop there is a guarantee to execute loop body minimum one time.

2) In case of while, first, conditional expression will be executed then body will be executed.

In case of do-while loop, first body will be executed, then conditional expression will be executed.

3) In case of while loop, conditional expression will be executed to perform current iteration.

In case of do-while loop, conditional expression will be executed to perform next iteration.

SYNTAX:-

```
do
do
{
    — instructions —
}
while (condition);
```

Ex:-①

```
int i=0;
do
{
    S.O.P(i);
    i=i+1;
}
while (i<10);
```

Status:- No C6

O/P:- 0 1 2 3 4 5 6 7 8 9.

Transfer Statements:-

These statements can be used to bypass flow of execution from one instruction to another instruction.

There are three transfer statements in java.

- 1) break
- 2) Continue
- 3) Return.

break:-

This transfer statement can be used to transfer flow of execution to outside of the loop by skip off the remaining instruction.

Ex:- ①

```
for(int i=0; i<10; i++)  
{  
    if(i==5)  
    {  
        break;  
    }  
    S.O.Pn(i);  
}
```

Status:- NO CE
OP:- 0 1 2 3 4.

(2) {

```
S.0.Pln("Before loop");
for (int i=0; i<10; i++)
```

{

```
S.0.Pln ("Inside loop");
if (i==5)
```

{

```
S.0.Pln ("Inside loop, before break");
break;
```

```
S.0.Pln ("Inside loop, after break");
```

{

```
S.0.Pln ("After loop");
```

{

Status:- Compilation Error, Unreachable Statement.

O/P:-

Reason:-

If we provide any statement immediately after break Statement then that Statement is unreachable Statement, where compiler will rise an error.

(3) {

```
for (int i=0; i<10; i++)
```

```
{ for (int j=0; j<10; j++)
```

```
{ if (j==5)
```

```
{ break;
```

```
S.0.Pln (i+ " " + j);
```

Status:- No CE.

i	j	i+j	i+j
0	0	20	40
0	1	21	41
0	2	22	42
0	3	23	43
0	4	30	44
1	0	31	50
1	1	32	51
1	2	33	52
1	3	34	53
1	4		

Note:- If we provide break statement inside nested loop then break statement will give effect upto the nested loop only, it will not give effect to Outer loop.

In the ~~above~~ context, if we want to give effect to a particular outer loop by keeping break statement in nested loop then we have to use "labelled break" statement.

Syntax:-

break label;



Ex:- (O)

```
l1: for( int i=0; i<10; i++ )  
    {  
        for( int j=0; j<10; j++ )  
            {  
                if (j==5)  
                    {  
                        break l1;  
                    }  
                System.out.println( i + " " + j );  
            }  
    }  
}
```

Status:- x10 CE

O/P:- 0 0

0 1

0 2

0 3

0 4



2) Continue Statement:-

This transfer statement can be used to bypass flow of execution to the next iteration by skipping current iteration.

ex: ①

```
for (int i=0; i<10; i++)
```

```
{ if (i==5)
```

```
    Continue;
```

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

Status:- NO CB

Op:- 0 1 2 3 4 6 7 8 9

(5 is skipping due to continue statement)

②

```
S.O.PA("Before Loop");
```

```
for (int i=0; i<10; i++)
```

```
{ S.O.PN("Inside loop");
```

```
if (i==5)
```

```
{ S.O.PN ("Inside loop, before Continue");
```

```
Continue;
```

```
S.O.PN ("Inside loop, after Continue");
```

```
{ S.O.PN (i);
```

```
{ S.O.PN ("After loop");
```

Status:- CB, Unreachable

Statement

③)

```
for (int i=0; i<10; i++)
```

```
{  
    if (j==5)
```

```
        for (int j=0; j<10; j++)
```

```
{  
    if (j==5)
```

```
        continue;
```

```
{  
    }
```

```
    System.out.println(i + " " + j);
```

```
{  
    }  
    Status:- 0 0
```

```
No of 0 1  
; ;  
; ;  
; ;  
9 9
```

If we provide continue statement inside nested loop then continue statement will give effect to nested loop only, not to the other loop. In this context, if we want to give continue statement effect to outer loop by keeping continue statement in nested loop then we have to use labelled continue statement.

SYNTAX:- Continue labelled;

Ex: ④)

```
l1: for (int i=0; i<10; i++)
```

```
{  
    for (int j=0; j<10; j++)
```

52

$\ell^j (j=5)$

{ Configure l_1 ;

{ S.O.P/M ($l+4$ $+j$) ;

{ Status:- No CE.

{ OP :- 0 0
 0 1
 0 2
 0 3
 0 4

~~=====~~

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

10-08-2015

OOPS (Object Oriented Programming Security)

Object Orientation:-

To design applications we have to use either of the following types of programming languages.

- 1) Unstructured Programming languages -
- 2) Structured Programming languages ✓
- 3) Object Oriented programming languages ✓
- 4) Aspect Oriented programming language ✓

Q: What are the differences between Unstructured programming languages and Structured programming languages?

→ Unstructured programming language are outdated programming languages, which are not suitable for at present application requirements.

Structured programming languages are not Out dated programming languages, which are suitable for Application requirement.

→ Unstructured programming language are not using any structure to design applications.

Structured programming languages are using a standard structure to design applications.

53

3) Unstructured programming languages are using mnemonic codes like ADD, SUB, MUL, ... to design applications which provide very less no. of features to design applications.

Structured Programming languages are using high level syntax to design applications which are available in more number and which will provide more no. of features to design applications.

④ Unstructured programming languages are using only goto Statement to define flow of execution. It is not sufficient for applications.

Structured programming languages are using more no. of flow controllers like goto, if, switch, for, while, ... to define flow of execution.

⑤ Unstructured programming languages are not using functions feature, it will reduce code reusability.

Structured programming languages are using functions feature, it will improve code reusability.

⑥ Examples for unstructured programming languages are BASIC and FORTRAN.

Example for Structured Programming languages are C and PASCAL.

Q What are the differences between Structured programming languages and Object Oriented Programming languages



1) Structured Programming languages are providing difficult approach to design applications.

Object Oriented programming languages are providing simplified approach to design applications.

2) Structured programming languages are providing less modularity.

Object Oriented programming languages are providing very good modularity.

3) Abstraction is not good in Structured programming languages

Abstraction is very good in Object Oriented programming language.

4) Security is not good in Structured programming languages.

Security is very good in Object Oriented programming languages.

5) Code reusability is not good in Structured programming languages.

Code reusability is very good in Object Oriented programming language due to "inheritance" kind of Object Oriented feature.

⑥ Sharability is not good in structured programming languages.

Sharability is very good in Object Oriented Programming languages.

⑦ Examples for structured programming languages are C and PASCAL.

Example for Object Oriented programming languages are C++ and JAVA.

→ Aspect Oriented Programming languages:-

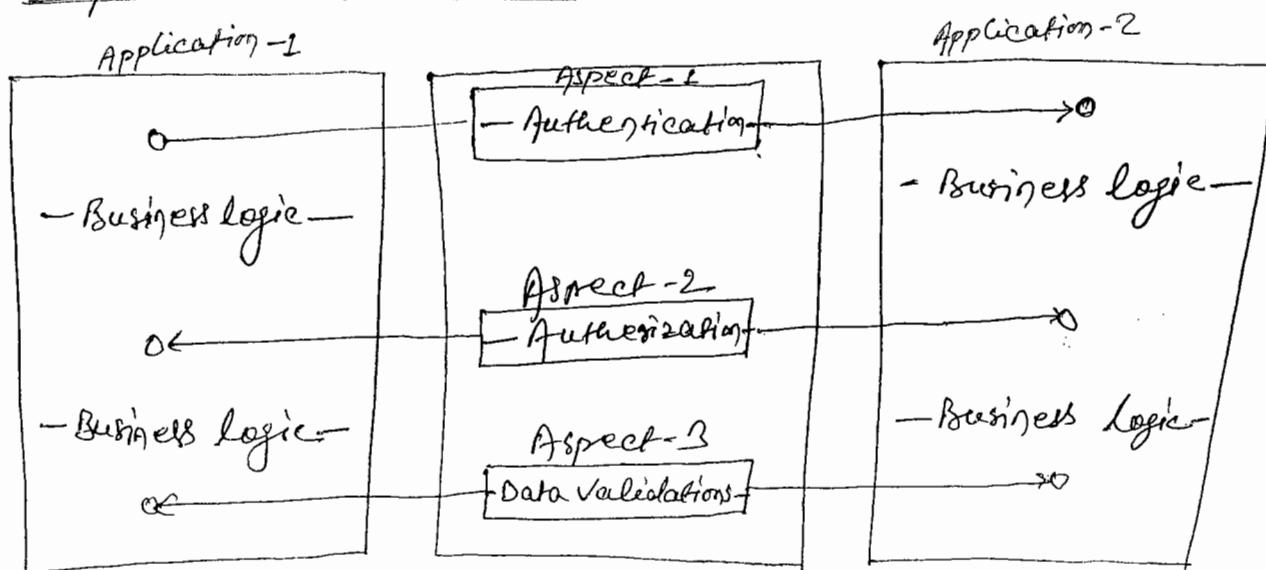
→ If we want to design applications by using OOPC, then we have to provide both services logic and business logic in combined manner, this approach will provide tightly coupled design, less sharability and less reusability.

Note:- In Enterprise applications, Authentication, Authorization, Security, Data Validation... are treated as services.

→ In the above context, if we want to improve loosely coupled design, more sharability and more reusability then we have to apply Aspect Orientation rules and regulation of Object Oriented Programming.

In Aspect Orientation, we have to separate all the services logic from business logic, we have to declare each and every service as an aspect and we have to

Inject aspects into enterprise applications at runtime
as per the requirement.



Note:- Aspect Orientation is a methodology, a set of rules and regulations applied on Object Oriented programming in order to provide more loosely coupled design, more sharability and more reusability. No programming language is existed on the basis of Aspect Orientation.

Note:- Aspect Orientation is very most useful in products developments like sever frameworks, ...

Ex:- Spring framework.

→ Object Oriented Features:-

To show the nature of Object Orientation, Object Orientation has provided the following features.

P.T.O

- 1) Class ✓
- 2) Object ✓
- 3) Encapsulation ✓
- 4) Abstraction ✓
- 5) Inheritance ✓
- 6) Polymorphism ✓
- 7) Message passing ✓

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

Q) the basis of the Object Oriented features, there are two types of programming languages

- 1) Object Oriented Programming languages
- 2) Object Based Programming languages.

Q) Difference between Object Oriented programming language and Object Based programming language.

⇒ Object Oriented programming languages are able to allow all the Object Oriented features including "Inheritance".

Ex:- Java.

Object Based Programming languages are able to allow all the Object Oriented features excluding "Inheritance".

Ex:- Java Script.

12-08-2015

Q. What are the difference between Class and Objects.



Class is a group of elements having common properties and common behaviours.

Object is any individual element among the group of elements having physical properties and behaviours.

2) Class is Virtual

Object is physical

3) Class is virtual encapsulation of properties and behaviours

Object is physical encapsulation of properties and behaviours.

4) Class is Generalization

Object is Specialization.

5) Class is a model or blue print for the objects.

Object is any instance of the class.

Q. What is the difference between Encapsulation and Abstraction?



The process of finding data and coding part as a single unit is called as "encapsulation".

The process of hiding unnecessary implementation and visualizing necessary implementation is called as "abstraction".

Note:- In java application, both encapsulation and abstraction are achieving "Security".

Inheritance:-

The process of getting variables and methods from one class to another class is called as "Inheritance".

The main objective of inheritance is code reusability.

Polymorphism:-

Polymorphism is a Greek word, where Poly means many, and Morphism means Structure.

If one thing is existed in more than one form then it is called as "Polymorphism".

The main objective of polymorphism is "flexibility" in application development.

Message Passing:-

The process of transferring data along with flow of execution from one instruction to another instruction is called as "Message passing".

In java applications, message passing is able to improve data navigation between entity classes and it is able to improve flow of execution.



Containers in java:-

There are three types of Containers in java.

1) Class

2) Abstract class

3) Interface

Container is a java element, if contains all the programming elements like variables, methods, constructors,

1) Class:-

The main intention of class in java applications is to represent real world entities in the form of coding part.

Ex:- Students, Employee, Customer, Account, . . .

Syntax:-

[Access-Modifiers-List] class class-name [extends Super-class] [implements Interface-List].

{

- Variables —
- Methods —
- Constructors —
- Blocks —
- Classes —
- abstract classes —
- Interfaces —
- enums —

}

W

Access Modifiers:-

There are two types of access modifiers in java.

1) To define scope to the programming elements like variables, methods, classes, ... We have to use the following access modifiers.

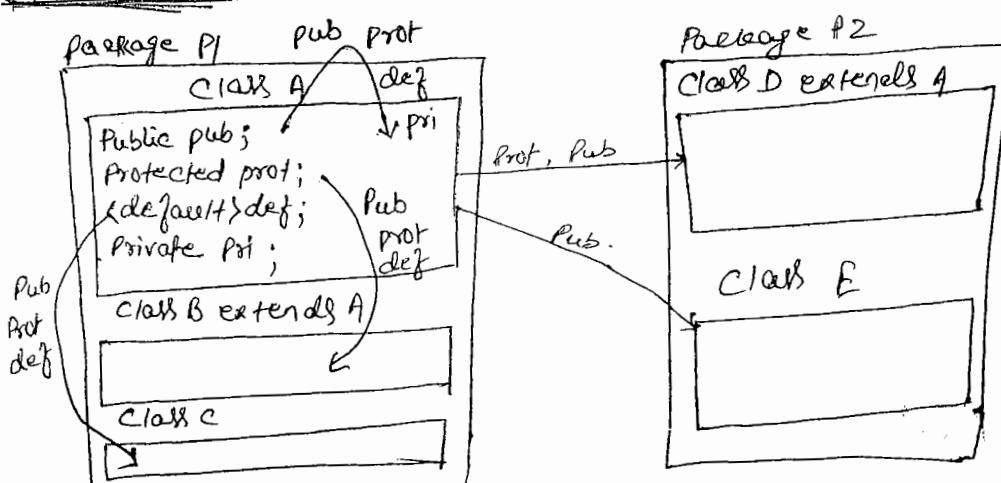
- Public —
- Protected —
- <default> —
- Private. —

Where "private" are available upto the respective class where private members are declared.

Where "<default>" members are available upto all the classes and interfaces available in the present package.

Where "Protected" members are available upto all the classes and interfaces available in the present package and the child classes available in the other packages.

Where "Public" members are available throughout the application.



from the above list of access modifier, Only public and default access modifiers are possible for classes, private and protect access modifiers are not possible for classes.

In the case of inner classes, all the scope related access modifiers like public, protected, default and private are allowed.

[Note:-

If any access modifier is define on the boundaries of classes then that access modifier is not applicable for the classes, that access modifier is applicable for members of the class excluding inner classes.

Ex:- In java private and protected scopes are defined on the boundaries of the classes, so that, they are not applicable for classes, they are applicable for members of the classes excluding inner classes, because inner class is a member of an outer class.

2) To provide some extra nature to the programming elements like variables, methods, ... We have to use the following access modifiers.

Static, final, abstract, native, volatile, transient, synchronized, Strictfp,

Applicable for Variable
Applicable for variable
Applicable for Method

From the above list of access modifiers, Only the access modifiers like final, abstract and strictfp are allowed for the classes, but inner classes are allowing static, final, abstract and strictfp access modifiers.

Note:- "Static" access modifier functionality is defined on the origin of classes, so that it can not be applied for classes, it can be applied for the members of the classes including inner classes, because inner class is a member in any outer class.

Ex:-

- 1) Public class A{} → Valid
- 2) Protected class B{} → Invalid ✓
- 3) Class C{} → Valid ✓
- 4) Private class D{} → Invalid ✓
- 5) Public class E{} Public class F{} → Valid
- 6) Class A{} Protected class B{} → Valid
- 7) Class A{} Private class B{} → Valid
- 8) Static class A{} → Invalid ✓
- 9) Class A{} Static class B{} → Valid ✓
- 10) Abstract class A{} Class B{} → Valid ✓
- 11) Final class A{} Class B{} → Valid ✓

14-08-2015

"> Strictly class A{} → Valid ✓

13) transient class A{} → Invalid ✓

→ Where "class" is a keyword, it able to represent
Class Object Oriented feature.

→ Where "className" is an identifier, it can be used to
recognize the class individually.

→ Where "extends" is a java keyword it can be used to
specify a particular super class in the class syntax in order
to get variables and methods from the specified super
class to the present sub class.

Note:- In java class syntax, "extends" keyword is able to
allow only one super class, it unable to allow
more than one super class. If we provide more than
one super class then that situation is representing multiple
Inheritance, it is not possible in java.

→ Where "implements" is a java keyword, it able to
allow one or more no. of interface names in order to
implement abstract methods.

Note:- In class syntax, extends keyword is able to allow
exactly one super class name but implements keyword
is able to allow more than one interface names.

Note:- In class syntax, both extends and implements
keywords are optional, we can write a class without
implements keyword, and we can write a class without
extends keyword.

both extends and implements keyword and we can write a class without both extends and implements keywords but if we want to provide both extends and implements keywords in class syntax then first we have to provide extends keyword then we have to provide implements keyword, it is not possible to interchange extends & implements keyword in class syntax.

- 1) Class A extends B {} → V ✓
- 2) Class A extends B, C {} → I ✓
- 3) Class A implements I {} → V ✓
- 4) Class A implements I1, I2 {} → V ✓
- 5) Class A implements I extends A {} → I ..
- 6) Class A extends B implements I {} → V ..
- 7) Class A extends I {} [I is interface] → I ..
- 8) Class A implements B {} [B is class] → I ✓

* Procedure to use Classes in java application:-

- 1) Declare a class by using "Class" keyword.
 - 2) Declare Variables and methods in the class as per the application requirement.
- Note:- In general, variables are able to represent data & methods are able to represent a particular action of an entity.

- 3) In main class and in main() method, create object for the respective class.
- ④ Access the required members of the class by using the generated reference variable.
- ~~Ex~~
- Class Student
- ```

String Sid = "S-999"; // Variables and methods (step 2)
String Sname = "AMARENDR A SINGH";
String Saddr = "Hyderabad";
String Semail = "amarendrakumar999@gmail.com";
String Smobile = "91-9471200315";
Public void displayStudent()
{
 S.O.Pln ("Student Details");
 S.O.Pln ("-----");
 S.O.Pln ("Student Id : " + Sid);
 S.O.Pln ("Student Name : " + Sname);
 S.O.Pln ("Student Address : " + Saddr);
 S.O.Pln ("Student Email : " + Semail);
 S.O.Pln ("Student Mobile : " + Smobile);
}

```
- if we want to take only Student() method, then it is possible or not?  
Ans → Yes, it is possible.
- Class Test
- ```

L PSVM (String[] args)

```

```

}
Student std = new Student();
std.display();
}
}

```

~~Methods:-~~ ^{OK}

In java there are two types of methods.

- ① Concrete methods ✓
- ② Abstract methods ✓

Q. What are the diff. between Concrete and Abstract method?

⇒ Concrete methods are the normal java methods, which must have both method declaration and method implementation.

Abstract methods are java methods, which must have only method declaration with out implementation part.

→ To declare Concrete method no special keyword is existed

To declare abstract method we must use "abstract" keyword.

→ Concrete methods are allowed in classes and abstract classes, but not allowed in interface.

Abstract methods are allowed in abstract classes and interfaces, but not allowed in classes.

→ Concrete methods are able to provide less sharability.

Abstract methods are able to provide more sharability.

Class A

void add(int i, int j) → Concrete method

```
{  
    System.out.println("Addition : " + (i+j));  
}
```

abstract class B

```
{  
    abstract void add(int i, int j) → abstract method.  
}
```

17-08-2015

Abstract Classes:-

Abstract classes are java classes, which are able to allow zero or more % of concrete methods and zero or more % of abstract methods.

[Note:- In java applications, we can declare abstract classes without concrete methods and with abstract methods, We can declare abstract classes without

abstract methods and with concrete methods, we can declare abstract classes without concrete methods and abstract methods.

Note:- In java applications, we can declare abstract classes without abstract methods, but to declare an abstract method then the respective class must be abstract class.

In java applications, for abstract classes we are able to declare reference Variables but we are unable to create objects.

In java applications abstract classes are able to provide more sharability when compared with classes.

Procedure to use abstract classes in java application:-

- 1) Declare abstract classes by using "abstract" keyword.
- 2) Declare Variables and methods in abstract class as per the application requirement.
- 3) Declare a sub class for the abstract class.
- 4) Implement all the abstract methods in sub class which are declared in the respective abstract class.
- 5) In main class, a main() method, Create object for the sub class and declare reference variable either for the abstract class or for the sub class.

6) Access the abstract class methods.

Note:- If we declare reference variable for abstract class
then we are able to access only the members which
are declared in abstract class, we are unable to access
sub class own members. If we declare reference variable
for sub class then we are able to access both abstract class
members and sub class own members.

Program:-

Abstract class A → we cannot create object for abstract class because it is undefined.
[abstract.java]

Void m1()
{

S.O.Println("m1 - A");
}

Abstract void m2();

Abstract void m3();

}

Class B extends A → Sub Class

{

Void m2()

{

S.O.Println ("m2 - B");
}

{

Void m3()

{

S.O.Println ("m3 - B");
}

{

Here, the meaning of ; (semi colon) is body is blocked.
It means that there is no body for
abstract method, because it is print
this method in other class, not in the
same class. So, it can't create the
object. But we can create the referen-
variable of it.

Q. What are the differences between classes and abstract classes.

1) Classes are able to allow only concrete methods, not possible to declare abstract methods.

Abstract classes are able to allow both concrete methods and abstract methods.

2) To declare classes no need to use any special keyword, along with "class" keyword, is sufficient.

To declare abstract classes we have to use 'abstract' keyword along with "class" keyword.

3) for classes we are able to declare both reference variables and objects.

for abstract classes we are able to declare Only reference variables, we are unable to create objects.

4) Classes will provide less sharability.

Abstract classes will provide more sharability.

Interfaces: OK

Interface is a java feature, it able to allow only abstract methods, there is no chance of declaring concrete methods.

in errors → { abstract → incomplete }
concrete → complete }

18-07-2015 6

for interfaces, we are able to declare reference variables but we are unable to create objects.

In case of interfaces, by default all the variables are public, static final.

In case of interfaces, by default all the methods are public and abstract.

Constructors are not allowed in interfaces. (because constructor is accessed by objects and objects are not created for interface) -
In Java applications, interfaces will provide more shareability when compared with class and abstract class.

Procedure to write Interfaces in java application:-

- 1) Declare an interface with ~~an~~ "interface" keyword.
- 2) Declare variables and methods in the interface as per the application requirement.
- 3) Declare an implementation class for the interface by including "implements" keyword.
- 4) Implement all the interface methods in the implementation class.

[Note:- While implementing interface method in Implementation class it is mandatory to declare with "public" access modifier, because, all the methods in interface are by default public.]

- 5) In main class, in main() method create object ~~implementer~~
for the Object class and declare reference
variable either for the interface or for the implementation
class.

- 6) Access the interface methods.

[Note:- If we declare reference variable of type interface then we are able to access only interface members, We unable to access implementation class own members. If we declare reference variable of type implementation class then we are able to access both interface members and implementation class own members.]

Programs:-

✓ Interface I

```

    {
        int x = 10;
        void m1();
        void m2();
        void m3();
    }
  
```

→ Interface. → there is no chance to create concrete method because in the interface methods are by default abstract.
variables & methods

Class A implements I → Implementation class

{ Public void m1()

{ S.O.P("m1-A"); }

{ Public void m2(); }

→ here, it is mandatory to declare public access modifier, because all the methods in interface are by default public. [Or, here method which is declared in interface and implement in implementation class are must be declare as public class are must be declare as public access modifier. Seez in interface the methods are by default public.]

```

        S.O.P("m2-a"),
    }
    {
        Public void m3()
    }
    {
        S.O.P("m3-a");
    }
    {
        Public void m4()
    }
    {
        S.O.P("m4-a");
    }
}

```

64

<u>OP:-</u>	10
	10
	10
	M1-A
	M2-A
	M3-A
	10
	M1-A
	M2-A
	M3-A
	M4-A

Class Test
 { → Main Class

{ Public static void main (String [args]

// I = new I(); → error → here, error occurred, because the methods

I = new A(); → Object created

S.O.P("I.x"); → Abstract method has no

S.O.P("A.x"); body. like Ext.

S.O.P("A.x"); Abstract class A

I.m1(); } Abstract void m2(); }

I.m2(); }

I.m3(); }

// I.m4(); Reference variables

A a = new A(); → Error → Here, error occurred, because m4() method is not declared in interface. So, we can't access m4() method by the reference variable of interface.

S.O.P("a.x"); → Object created of interface.

a.m1(); }

a.m2(); }

a.m3(); }

a.m4(); } Reference variables.

19-08-2015

Q. What are the differences between classes, abstract classes and Interfaces?



1) Classes are able to allow only concrete method.

Abstract classes are able to allow both concrete and abstract method.

Interfaces are able to allow only abstract methods.

2) For classes only, we are able to provide both reference variables and objects.

For abstract classes and interfaces we are able to provide only reference variables, we are unable to create objects.

3) In case of Interfaces, by default, all the variables are "public static final".

In case of classes and abstract classes, no default cases for variables.

4) In case of Interfaces, by default, all the methods are "public and abstract".

In case of classes and abstract classes, no default cases are available for methods.

5) Constructors are possible in classes and abstract classes.

Constructors are not possible in interfaces.

6) Classes will provide less sharability.

Abstract classes will provide middle level sharability.

Interfaces will provide more sharability.

Methods in java:-

The main intention of the methods in java programming is to represent actions/Behaviours of an entity which is represented in the form of a class.

Syntax:-

```
[Access - Modifiers - List] return - Type method_Name ([Param - List])
[throws Exception - List].
{
    _____ implementation _____
}
```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

Java methods are able to allow the access modifiers like public, protected, (default) and private one at a time.

Java methods are able to allow the access modifiers like static, final, abstract, native, synchronized, Strictfp.

→ As Java is a typed programming language we have to specify a particular type to represent the type of data which we want to return from methods, for this we have to use return type.

In Java applications, Java methods are able to allow all primitive data types, all user defined data types and void of return type.

Note:- "Void" return type by representing no data is returned from the current method.

- Where method name is an identifier, it can be used to recognize few methods individually.
 - Where method parameter list can be used to get input data to the methods in order to perform the respective action which we implemented inside the method.
 - Java methods are able to allow all primitive data types and all user defined data types as parameters, but not "void".
 - Where "throws" keyword can be used to the generated exception from present method to the caller method.
- [Note:- throws keyword is able to allow more than one exception class.]

Representing the Methods:-

There are two ways to describe methods.

- 1) Method Signature
- 2) Method Prototype

Q: What is the differences b/w method signature and method prototype.

⇒ Method Signature is the description of a method including method name and method parameters list.

Ex:- forName(String className) -

Method prototype is the description of a method excluding access modifiers list, method return type, method name, parameter list and throws exceptions list.

Ex:-

public Class.forName(String class-name) throws ClassNotFoundException

There are two types of methods in java

- 1) Mutator Methods → Use to set the data to the object
- 2) Accessor Methods → Use to get the data from the object

Q What is the difference between Mutator method and Accessor method?

→ Mutator method is a java method, it can be used to set data to the object.

Ex:- All SetXXX() methods in java bean classes are mutator methods.

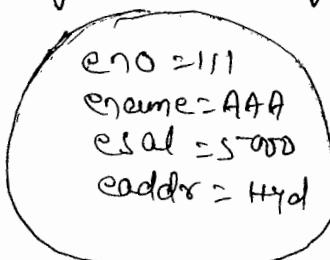
Accessor method is a java method, it can be used to get data from the object

Ex:- All getXXX() methods in java bean classes are accessor methods.

Mutator Methods:

```
e.setEno(111);
e.setEname("AAA");
e.setEsal(5000);
e.setEaddr("Hyd");
```

```
Employee e=new Employee();
```



Accessor Methods

```
e.getEno(); 111
e.getEname(); AAA
e.getEsal(); 5000
e.getEaddr(); Hyd.
```

Note:- java bean is a normal java class, it contains properties and the respective SetXXX() methods and getXXX() methods in order to manage entities data.

20-08-2015

If we want to write java bean classes, we have to use the following rules and regulations.

- 1) Declare a non-abstract Class (concrete class)
- 2) Declare all the variables as private.
- 3) Declare a separate set of setxxx() method and getxxx() method for each and every property.
- 4) Declare all the methods in java bean classes as public.
- 5) If we want to provide any constructor in java bean class then provide a constructor that must be 0-arg constructor and non-private constructor.

Programs:-

Class Employee → Non-abstract / Concrete Class
{
 private int eno; → These are class level variable / Globalized variable.
 private String ename; → Declaring all the variables as private.
 private float esal;
 private String eaddr;
 public void seteno(int eno) { → Declared a separate setxxx() method
 eno = eno; → for each and every property / variable
 }
 public void setename(String ename) { → which we declared in class
 ename = ename;
 }

```

public void setSal (float esal1)
{
    esal = esal1;
}

public void setEaddr (String eaddr1)
{
    eaddr = eaddr1;
}

public int getEno() → declare a separate getyy() method
{
    return eno;
}

public String getName()
{
    return ename;
}

public float getSal()
{
    return esal;
}

public String getEaddr()
{
    return eaddr;
}
}

```

Class Test → Main class

```

public static void main (String [] args)
{
    Employee emp = new Employee ();
    emp.seteno (111);
    emp.setename ("AAA");
    emp.setsal (5000.0f);
}

```

Employee emp = new Employee (); → Object created for Con create Class (class Employee).

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

```
emp.setFaddr("Aad");
S.o.println("Employee Details");
S.o.println("____");
S.o.println("Employee Id :" + emp.getEno());
S.o.println("Employee Name :" + emp.getEname());
S.o.println("Employee Salary :" + emp.getEsal());
S.o.println("Employee address :" + emp.getFaddr());
}
}
```

Variable-Argument Methods

In java application, if we define any method with "n" no. of parameters then we are able to access that method with the same "n" no. of parameter values. We are capable to access that method by passing "n-1" no. of parameter values and "n+1" no. of parameter values.

As per the requirement, we want to define a method and we want to access that method by passing variable no. of parameter values. To achieve this requirement, JDK5.0 version has provided a new feature called as "Variable-Argument Method", in short, "Var-Ag" method.

Var-Ag Method is a normal java method, it must have Var-Ag parameter.

Syntax:-

```
void m1(data-Type ... a) {
    —— implementation ——
}
```

When we access Var-arg method with parameter values then var-arg parameter is converted as an array of the Specified data Type and it will store all the provided Parameter Values.

Programs:-

Class A

```
void add(int... a) // int[] a (..., ..., ...)

S.O.Pln ("No of Arguments : " + a.length);
S.O.Pln ("Argument values : ");
int result = 0;
for (int i=0; i<a.length; i++)
{
    S.O.Pln (a[i] + " ");
    result = result + a[i];
}
S.O.Pln ();
S.O.Pln ("Addition : " + result);
S.O.Pln ("-----");
}
```

20-08-2015

Class Test → Main Class

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

A a = new A(); → Object Created

a.add(1);
a.add(10);
a.add(10, 20);
a.add(10, 20, 30);

This keyword can
not used inside
the static method

}

21/08/2015

In Var-Ag Method, it is possible to provide normal parameters along with var-arg parameters, but they must be provided before var-arg parameters, not after var-arg parameter, because in var-arg method var-arg parameter must be last parameter.

Note: Due to the above reason it is not possible to provide more than one var-arg parameter within a single var-arg Method.

Ex:-

void m1 (int ... a) {} → Valid ✓

int · ① m1 ... int

{ int ... a, float ... b }
(int a, float ...)

void m1 (int ... a, float ...) {} → Invalid ✓

void m1 (float f, int ... a) {} → Valid ✓

void m1 (int ... a, float ... f) {} → Invalid ✓

Object Creation in java:-

The main requirement to create objects in java application is,

- ① To store entities data temporarily.
- ② To access the members of a particular class.

To create objects in java application then we have to use the following syntax:

```
Class-Name ref-Var = new Class-Name((Param - List));
```

Ex:-

```
Class A {  
    int i;  
    float f=22.2f;  
    char c;  
}  
A a;  
}
```

```
A a=new A();
```

When JVM encounter the above extracting, JVM will create an object for the specified class with the following steps.

- ① Creating Memory
- ② Generating Identifiers
- ③ Providing initial values.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

① Creating Memory:-

When "JVM" encounter "new" keyword, JVM will load the respective class bytecode to free memory [Method Area] and JVM will recognize all the instance variables of the loaded class and identified memory size for the object on the basis of the instance variables data types.

After getting memory size details, JVM will send a request to heap manager about to create an object, where heap manager will search for the required block of memory in Heap memory and allocates required block memory for JVM as an object to store the respective class data.

② Generating Identities:-

After creating memory for JVM request, heap manager will assign an unique identity in the form of an integer value called as "HashCode".

Note:- To Create a HashCode value, heap manager will use its algorithm internally, where one of the factor was considered as memory allocation of the object -

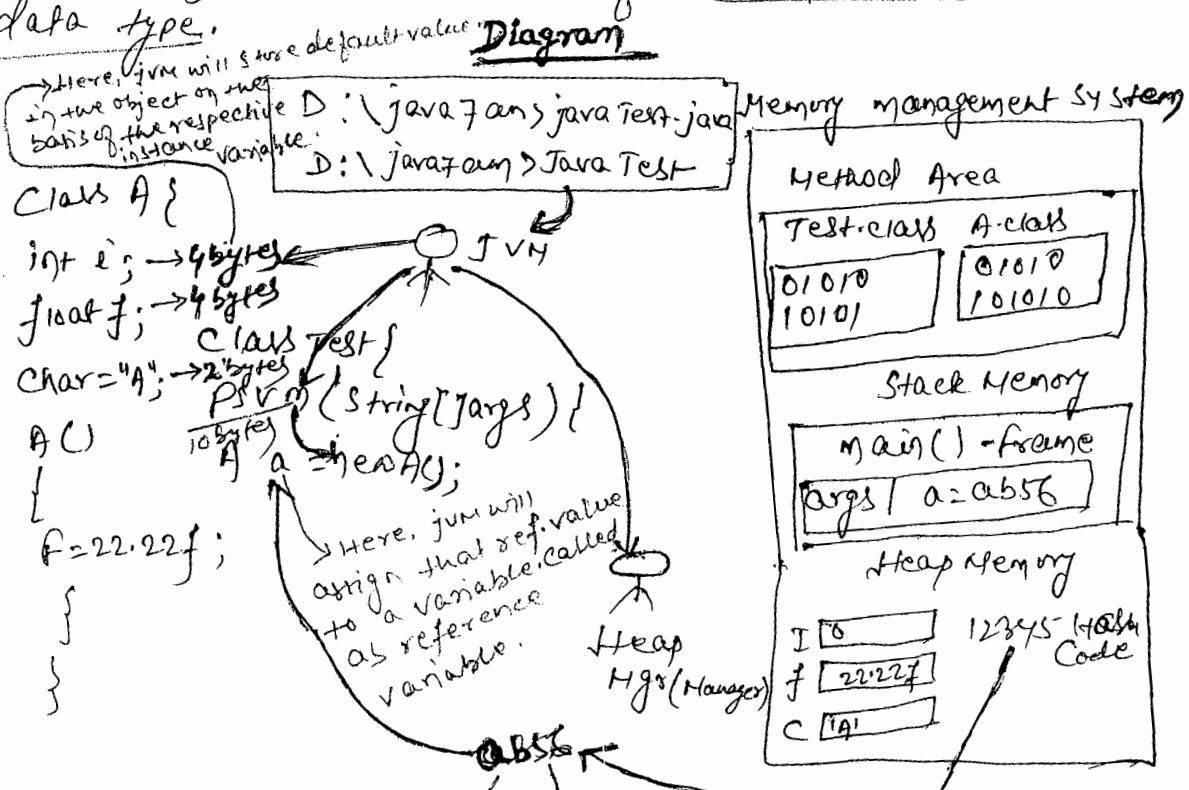
After getting Hashcode value, heap manager will send that Hashcode value to JVM, where JVM will convert that Hashcode value to its Head Decimal form called as Reference value of the object. After getting reference value, JVM will assign that reference value to a variable called as

Reference Variable:

③ Providing initial values to the Object:-

After generating identity for the Object, JVM will allocate memory for all the instance variables of the respective class in the Object and JVM will provide initial values to the instance variables inside the object by searching its class level declarations and at the provided Constructors.

If any variable is not having initialization at class level declaration and at constructor then JVM will store default value in the object on the basis of the respective instance variable data type.



JVM will convert the HashCode value to the Hexadecimal. It is Hexadecimal. It is called as reference value.

If we want to get hashCode value of an object then we have to use the following method.

public native int hashCode()

Note:- If any method declared in Java but implemented in Non-Java programming languages like C, C++, ASL, HW languages, then that method is called as Native method.

→ hashCode() method declared in Java but implemented in HW language in order to get Object hashCode value.

If we want to get Object reference value of an object then we have to use the following method:

public String toString()

Note:- toString() method will reference value of an object in the following format in the form of a String.

Class-Name@Ref-Val

Class A {

}

Class Test {

}

PSVM (String args) {

 A a = new A();

 int hc = a.toString();

 int hc = a.hashCode();

 String ref = a.toString();

 S.O.P19 ("HashCode : " + hc);

 S.O.P19 ("Ref Value : " + ref);

}

Ques:-

- ① hashCode :- 26022015
 Ref value :- A@18d107f
- ② hashCode :- 3541984
 Ref value :- A@3605e0

It will be varied by
time to time.

In java programming language, `java.lang.Object` class is common and default super class for every java class, here java classes may be predefined classes or User defined classes. In `java.lang.Object` class there are 11 no. of methods in order to reuse at each and every java class.

- hashCode() ✓
 - toString() ✓
 - equals(Object obj) ✓
 - getClass() ✓
 - finalize() ✓
 - wait() ✓
 - wait(int time) ✓
 - wait(int time, int time) ✓
 - notify() ✓
 - notifyAll() ✓
 - clone() ✓
- ≡

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road.
 Ameerpet, Hyderabad.

23-08-2015

Q Consider the following code, where Class B is having explicit super class A and implicit super class that is `java.lang.Object`, it shows clearly multiple inheritance, how can we say multiple inheritance is not available in java?

Class A

{

}

Class B extends A

{

}

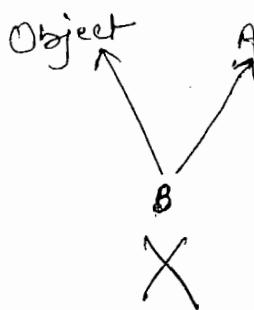
→ In java applications, `java.lang.Object` class is a default super class for the java class whenever the respective java class is not extending any other class explicitly, if our class is extending other class explicitly then `java.lang.Object` class is not directly super class, it is super class for the class through multi level inheritance [through explicit super class], not through multiple inheritance.

Class A {

}

Class B extends A {

}



///

In java applications when we pass reference variable as parameter to `System.out.println()` method then JVM will access `toString()` method internally.

Class A.

}

Class Test -

{

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

{

`A a = new A();`

`String ref = a.toString();`

`S.0.Println(ref); // A@abs6`

`S.0.Println(a.toString()); // A@abs6.`

`S.0.Println(a); // A@abs6`

}

}

=

*Note:-
Object created for String = object.toString.
String ref = &.toString();*

When we access `toString()` explicitly or implicitly, JVM will search for `toString` methods in the respective class, where if `toString()` method is not available then JVM will search for `toString()` at super class, if no super class is existed explicitly then JVM will access `java.lang.Object` class `toString()`, it was implemented in such a way that to return a String contains "Class-Name@Ref-value".

In java applications, if we pass object reference variable as parameter to ^{SOPIN(-)} ~~the method~~ then JVM will execute Object class `toString()` method, where Object class `toString()` method will return a String contains, Class-Name@Ref-val, but, as per the requirement, we don't want to display reference value of the object, where we want to display some other details which are required by the application.

To achieve the above requirement, we have to define our own `toString()` method in the respective class.

Ex:-

Class Employee -

{

String eid = "E-111";

String ername = "Durga";

String eemail = "durga@durgasoft.com";

String emobile = "91-9471200315";

↳ ^{Format} ^{String to Primitive}
→ toString

gethr

getdata

get...
get...
get...
get...

public String toString() -

S.OPIN ("Employee Details") is - "from return" "We can
S.OPIN ("-----"); store all the data in String.
S.OPIN ("Employee Id :" + eid);
S.OPIN ("Employee Name :" + ername);
S.OPIN ("Employee Email :" + eemail);
S.OPIN ("Employee Mobile :" + emobile);
return " "; → String return by " ". Here we can return only string
value!

```

}
}

class Test {
    public static void main(String[] args) {
        Employee e = new Employee();
        System.out.println(e);
    }
}

initial thread id = 0
default value three = 5

```

~~Employee~~

~~System.out.println(e);~~

~~class~~

~~String~~

```

class Test {
    public static void main(String[] args) {
        String str = new String("Durga Software Solutions");
        System.out.println(str);
        Exception e = new Exception("My Own Exception");
        System.out.println(e);
        Thread t = new Thread();
        System.out.println(t);
    }
}
```

~~Thread~~

Note:- In java some predefined classes like String, StringBuffer, Exception, Thread, ... are not using object class `toString()` method. They are using their own `toString()` methods in order to display their own data.

O/P:-

Durga Software Solutions.

`Java.lang.Exception`: My own Exception

`Thread` [Thread - 0, 5, main] → Initial thread identities → 0
Default value of thread → 5
→ for execute from main method.

Types of Objects:-

* There are two types of objects in java:-

1) Immutable Objects

2) Mutable Objects.

Q. What are the differences between Immutable and Mutable Objects?

→ Immutable Object is a java object, it will not allow modifications on its content, data is allowed for the operations, but the resultant data will not be stored back in the original object, the resultant data will be stored by creating new object.

Ex:- String class objects are immutable objects

All wrapper class objects are immutable objects.

→ Mutable object is java object, it able to allow modifications directly on its content.

Ex:- By default, all java objects are mutable objects, except String class objects, Wrapper class objects, ...

Ex:- StringBuffer (it is an example of Immutable object)

Ex:- (Immutable Objects)

Class Test → Main Class

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

```
    String Str1 = new String ("Durga");
```

```
    String Str2 = Str1.concat ("Software");
```

```
    String Str3 = Str2.concat ("Solutions");
```

```
    System.out.println (Str1);
```

```
    System.out.println (Str2);
```

```
    System.out.println (Str3);
```

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

```
StringBuffer Sb1 = new StringBuffer ("Durga");
```

```
StringBuffer Sb2 = Sb1.append ("Software");
```

```
StringBuffer Sb3 = Sb2.append ("Solutions");
```

```
System.out.println (Sb1);
```

```
System.out.println (Sb2);
```

```
System.out.println (Sb3);
```

Durga

Str1[0111]

Durga Software

Str2[02222]

Durga Software Solutions

Str3[0333]

Durga

Durga Software

Durga Software Solutions

Sb1

[d444]

Sb2

[d444]

Sb3

[d444]

Final Output:

Durga

Durga Software

Durga Software Solutions

Durga Software Solutions

Durga Software Solutions

Durga Software Solutions

Q What's the difference between Object and instance?

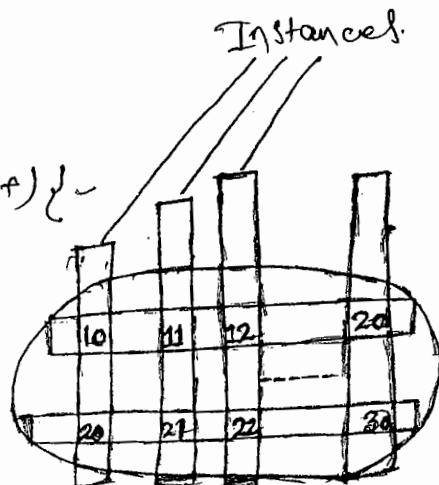


Object is a memory element to store data.

Instance is a copy of values or layer of values at a particular point of time in an object.

Note:- In java applications, single object may have no. of instances but always we are able to get the latest or top most instance, not previous instances.

```
Class A} ~ Class Test{ ~  
int i=10; ~ ~ PSVM (---){ ~  
int j=20; ~ ~ A a=new A(); ~  
} ~ ~ for (int x=0; x<10; x++) { ~  
~ ~ a.i = a.i+1; ~  
~ ~ a.j = a.j+1; ~  
~ } ~ ~ } ~ ~ }
```



i [10 11 12 --- 20]

x
j [20 21 22 --- 30]

Instance Context:-

In java applications, for each and every class loading a separate context will be created called as "Static Context".

In java applications, for each and every Object, a separate context will be created called as "Instance Context".

In java, Instance Context is represented in the form of the following element.

- ① Instance Variable
- ② Instance Method
- ③ Instance Block.

① Instance Variable:-

→ Instance Variable is a java variable, it will be recognized and initialized just before executing the respective class constructor.

→ Instance Variable is a normal java variable, its values are changed from one instance to another instance of an object.

→ In java applications, instance variables must be declared as class level variables, not as static variables and not as local variables.

→ In java applications, Instance Variables data is stored in the form of objects in heap memory.

② Instance method:-

→ It is a normal java method, it will represent a particular action.

→ In java applications, instance methods are recognized and executed the moment when we access that instance method.

→ In java applications, all non-static methods are treated as instance methods.

Class A

}

int i = m1(); ①
A(); ②
} ③
constructor

S.O.P("A-con");
}

int m1() point 2

{
S.O.P("m1-A"); point 1
return 10;
}

}

Class Test → Main class

{
PSVM (String [] args);

{
A a = new A(); object created.
}

OP:- m1-A
A-con

Instance Block :-

It is a set of instructions, it will be recognized and executed just before executing the respective class constructor.

Syntax :-

```
{  
    --- instructions ---  
}
```

Ex:-

```
Class A -  
{  
    A () -  
    {  
        S.O.P("A-con"); -  
    }  
    {  
        S.O.P("IB-A"); -  
    }  
}  
Class Test -  
{  
    Psvm (String[] args) -  
    {  
        A a = new A (); -  
    }  
}
```

O/P:- IB-A
A-con

In a java class, if we provide instance variables, instance methods and instance block then JVM will execute all those elements before executing the constructor in an order in which we have provided from starting point of the class to ending point of the class.

```
Class A ~
{
    A() ~> Constructor
    {
        S.O.P("A-con");
    }
    ejt m1() ~> Non static method
    {
        S.O.P("m1-A");
        return 10;
    }
    ejt e=m1(); ~> Non static variable
    {
        S.O.P("IB-A");
    }
}
Class Test ~
PSVM (String []args) ~
{
    A a = new A();
}
```

Static Block
Non St

O/P:-	m1-A
	IB-A
	A-con

Constructor:-

- Constructor is a java feature, it can be used to construct the object.
- The role of the constructors in Object Creation is to provide initial values in the Object.
- In java applications, constructors are executed exactly at the time of creating objects, not before creating objects and not after creating objects.
- In java applications, constructors are used (utilized) to provide initializations to the class level variables.
- Constructors must have the same name of the class.
- Constructors are not having return types.
- Constructors are not allowing the access modifiers like static, final, abstract, ...
- Constructors are allowing the access modifiers like public, protected, (default), private.
- Constructors are allowing 'throws' keyword to bypass the generated exception from present constructor to the caller of the constructor.

Syntax:-

[Access - Modifier] Class - Name ([Param - List]) [throws - Exception - List]
 }

{

25-08-2015

Note:- If we declare constructor without the same class name and with different type then compiler will rise an error like "Invalid method declaration, return type required," because compiler has treated the provided constructor as ~~a~~ normal java method without the return type. In java methods, return type is mandatory.

Note 2:- In java apps, if we provide return type to the constructors then that constructor is converted as a normal java method, it will be executed when we access like normal java method.

Eg:-

Class A
{
 Void A()
 {
 System.out.println("A-con");
 }
}

Class Test
{
 Public static void main(String[] args)
 {
 reference variable
 A a = new A();
 object created
 a.A();
 reference variable
 }
}

→ Here, We can't provide return type to the constructor, if we provide then it will be converted into a normal java method, then it will be executed when we access as a normal java method.

→ Here Constructor call back by reference variable and we must have to create object for call back the constructor.

O/P:- A-con

constructor call back here.

Note:- In java applications Constructors are got allowing the access modifiers like, static, final, abstract, ... but if we provide these access modifiers to constructors then compiler will rise an error like "Modifier XXX not allowed here".

Ex:-

```
Class A {
    static A() { } // Here, constructor with access modifiers (static).
    S.O.Pn ("A-con"); // the access modifiers like static, final, abstract, ...
} // are not allowed in constructor, so, that there is
  // "Compilation Error" Occurred.
```

States:- Compilation Errors.

Note 4:- In java applications, Constructors are allowing the access modifiers like public, protected, <default> and private. If we declare Constructor as private then that constructor must be accessed within the same class only, it's not possible to access that constructor in outside of that class.

Ex:-

```
Class A {
    private A() { } // Here, constructor with Access modifiers (private)
    S.O.Pn ("A-con"); // and the access modifier like, private, public,
} // <default>, protected are allowed in the constructor.
  // If we declare constructor with this Access Modifier
  // then constructor must be accessed within the same
  // class only, it is not possible to access that constructor
  // in outside of that class.
  // And here, the constructor is accessed by the main
  // class, So, here Compilation error will be occurred.
```

Class Test

P.Svm (String[] args)

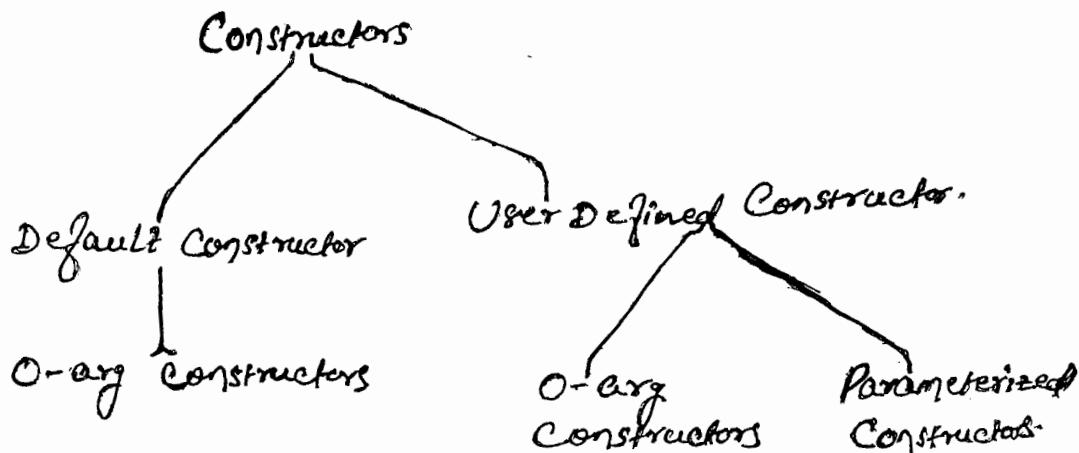
A a = new A();

}

Call back to constructor and here it'll call
by Main class so definitely compilation error
will occur.

States:- Compilation Errors.

There are two types of constructors in java.



Default Constructor:-

It is a normal constructor provided by the compiler when we are not declaring any constructor in java class explicitly.

- If we declare any constructor explicitly in a class then compiler will not provide default constructor.
- In java applications, compiler will provide only 0-arg constructors as default constructors, compiler will not provide parameterized constructor as default constructor.
- In java applications, compiler will provide default constructors with the same class scope.

[Note:- In java applications, default constructors are provided by the compiler. Only 0-arg by Jav.]

Ex:- D:\Java\test.java

public class Test

{
}

O₁ Command prompt:-

D:\javatam>javac Test.java.

D:\javatam>javap Test

Compiled from Test.java

public class Test extends java.lang.Object

{
 public Test(); → **Default Constructor** provided by compiler
 not by developer
 so, it's default, not user-defined.

Note:- 'javap' command can be used to display all the declarations in a particular class on command prompt.

User Defined Constructors:-

These Constructors are defined by the developers as per their application requirements.

There are two types of User Defined Constructors.

1) 0-arg Constructors

2) Parameterized Constructors.

1) 0-arg Constructors:-

If we declare any constructor explicitly in java class without having any parameter then that constructor is called as 0-arg constructor.

Note:- In java, all default constructors are 0-arg constructors but all 0-arg constructors are need not be default constructors.
Some 0-arg constructors provided by compiler are default constructors and some other 0-arg constructors provided by developers are called as User defined Constructors.

2) Parameterized Constructors:-

These constructors are defined by the developers with atleast one parameter.

✓ Class Employee

Parameter.java

```
String eid;
String ename;
float esal;
String eaddr;
```

Non-Static Variable

Non Static Variable
↓ access
constructor
↓
new object

Employee() → constructor with 0-arg.

{

```
eid = "E-111";
ename = "Durga";
esal = 50000.0f;
eaddr = "Hyd";
```

static
↓ access
class name

constructor with
accept modifier (public)
for getting Employee details.
we create it for static
object and point to
access this.

public void getEmpDetails()

⇒ In static method
we can not use
this.

so access by
reference of
new object

```
S.O.P("Employee Details");
S.O.P("-----");
S.O.P("Employee Id :" + eid);
S.O.P("Employee name :" + ename);
S.O.P("Employee salary :" + esal);
S.O.P("Employee Address :" + eaddr);
```

→ Constructor call back by
object

Class Test

reference variable.

```
psvm (String [ ] args)
Employee emp = new Employee();
emp.getEmpDetails();
```

In the above program, if we create more than one object for Employee class by using the provided 0-arg constructor then at all the objects the same data will be stored.

→ As per the appl' requirement, if we want to create more than one object of the same class with different data, then we have to use parameterized Constructors.

Ex:-

Class Employee → printemp.java

```
String eid;
String ename;
float esal;
String eaddr;
```

} → Non-Static Variables.

Employee (String eid, String ename, float esal, String eaddr) → Constructor with parameters

{ eid = eid;

parameterized constructor

ename = ename;

esal = esal;

eaddr = eaddr;

}

public void getEmpDetails() → Here Constructor with Access Modifiers (public)
for getting Employee details
we create it for return the details
by object and print.

S.O.P("Employee Details");

S.O.P("-----");

S.O.P("Employee Id :" + eid);

S.O.P("Employee Name :" + ename);

S.O.P("Employee Salary :" + esal);

S.O.P("Employee Address :" + eaddr);

}

Class Test

PSUM (String [args])

}
Employee emp1 = new Employee ("E-111", "Durga", 5000.0f, "Hyd");
emp.getEmpDetails();
S.O.P() ;
Employee emp2 = new Employee ("E-222", "Supriya", 6000.0f, "Hyd");
emp.getEmpDetails();
S.O.P();
Employee emp3 = new Employee ("E-333", "Ashoka", 7000.0f, "Hyd");
emp.getEmpDetails();
S.O.P();

}

Object 1
emp1
cid = E-111
ename = Durga
esal = 5000
eaddr = Hyd

Object 2
emp2
cid = E-222
ename = Supriya
esal = 6000
eaddr = Hyd

Object 3
emp3
cid = E-333
ename = Ashoka
esal = 7000
eaddr = Hyd

Here three object created
for three Employees detail

≡

Constructor Overloading:-

If we declare more than one constructor with the same name and with the diff. parameter list then it's called as Constructor Overloading.

✓ Class A → constructor - java

{ int i, j, k;

A() → constructor with 0-arg.

}

↳ A(i) → parameterized
constructor with one parameter.

{
i = i1;

}

A(i1, j1) → parameterized
constructor with two parameters

{
i = i1;

j = j1;

}

A(i1, j1, k1) → parameterized constructor with three parameters

{
i = i1;

j = j1;

k = k1;

}

public void add() → constructor with Access modifier (public)
and it must be in class level and here
it is. So, it's possible. Here public void add() method is used for addition
purpose. We create it for return the add and point it.

{
System.out.println("Addition :" + (i + j + k));

}

Class Test

PSVM (String[] args)

{
A a1 = new A(); → This object will call the A() constructor.
a1.add(); → 0
A a2 = new A(0); → This object will call the A(int i) constructor.
a2.add(); → 10
A a3 = new A(10, 20); → This object will call the A(int i, int j),
a3.add(); → 30
A a4 = new A(10, 20, 30); → This constructor will call the A(int i, int j, int k) constructor;
a4.add(); → 60
}
}=

Here, we create four objects for a class which is mentioned above.

call back to outer constructor

'this' keyword:-

'this' is a java keyword, it able to represent current class object.

→ In java applications, we are able to utilize 'this' keyword in the following four ways.

- 1) To refer current class variables.
- 2) To refer current class Constructors.
- 3) To refer current class methods.
- 4) To ~~refer~~ return current class Object.



3) To refer Current class Variables:-

If we want to refer current class variables by using 'this' keyword then we are able to use the following Syntax:

`this.var-name;`

Note: In java class, when we have same set of variables at local and at class level, where to access class level variables over local variables then we have to use 'this' keyword.

✓ Class A

```

    {
        int i=10; } → Global variable or, class level variable
        int j=20; } → Non static variable.

        A(int i, int j) → parameterized constructor. printed
    }

    S.O.P(i+ " " +j); → Local level variable printed
    S.O.P(this.i+ " " +this.j); → 10,20 → Class Level variable printed
    {   Here, if we not used 'this' keyword then
        }   Only one obj will be showed which
    }   is 100,200 and if we not consider
    }   S.O.P(i+ " " +j); then only one object is
    }   showed which is 10 20.
  
```

Class Test

```

    {
        PSVM(String[]args) →
    }

    A a=new A(100,200);
    }
  
```

O/P:- 100 200
10 20

27-08-2015

In java bean classes, we have to write setxxx() methods to set data to the variables inside the objects. In setxxx() methods we have to declare local variable name same as class level variables name and we have to transfer data from local variables to class level variables. In this context, to refer class level variables over local variables we have to use this keyword.

```
class Employee
{
    String eid;
    String ename;
    float esal;
    String eaddr;

    public void seteid(String eid)
    {
        this.eid = eid;
    }

    public void setename(String ename)
    {
        this.ename = ename;
    }

    public void setesal(float esal)
    {
        this.esal = esal;
    }

    public void seteaddr(String eaddr)
    {
        this.eaddr = eaddr;
    }

    public String geteid()
    {
        return eid;
    }
}
```

```

}
public String getName()
{
    return empname;
}

public float getSal()
{
    return esal;
}

public String getAddress()
{
    return eaddr;
}
}

```

Class Test

```

}
public void main(String[] args)
{
    Employee emp = new Employee();
    emp.setId("E-11");
    emp.setName("AAA");
    emp.setSal(5000.0f);
    emp.setAddr("Hyd");

    System.out.println("Employee Details");
    System.out.println("-----");
    System.out.println("Employee Id : " + emp.getId());
    System.out.println("Employee Name : " + emp.getName());
    System.out.println("Employee Salary : " + emp.getSal());
    System.out.println("Employee Address : " + emp.getAddress());
}
}

```

for set the value of variables
reference variable
emp.getId() will give the output

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

② To Refer Current class Constructor:-

If we want to refer current class constructor by using "this" keyword then we have to use the following syntax.

`this([Param-list]);`

Ex:-

Class A

{

AC()

{

this(10);

[Note:- It must be the first statement, otherwise compilation error occurred.]

S.0.PIN ("A-con");

}

A (int i)

{

this(10.0f)

S.0.PIN ("A-float-param-con");

}

A (float f)

{

this(22.2f)

S.0.PIN ("A-double-param-con");

}

A (double d)

{

S.0.PIN ("A-double-param-con");

}

S.0.PIN ("A-double-param-con");

It shows integer, means it should jump on int i

lastly this "A-con" will be printed.

It shows float means it should jump on float f

3rd print this int-param
After that it should jump back on previous

double, means it should jump on double

second print this float-param
After that it should jump back on previous.

here first print this double
after that it can jump back on previous

Class Test

```
PSVM (String [] args)
{
    A a = new A();
}
```

```
{}
=
```

O/P:- A-double-param-con
 A-float-param-con
 A-eigt-param-con
 A-con.

If we want to refer current class constructor by using "this"
Statement then the respective "this" Statement must be the
first Statement, if it's not first Statement then compiler will
give an error.

Class A

```
{
```

```
A()
```

```
{
```

```
S.O.P("A-con");
```

```
this(10);
```

```
}
```

```
A(eigt e)
```

```
{
```

```
S.O.P("A-eigt-param-con");
```

```
this(10.0f);
```

```
}
```

```
A(float f)
```

```
{
```

```
S.O.P("A-float-param-con");
```

```
this(22.22);
```

CE, because of that Statement only, it must
 be the first statement, here it is not first
 statement, so that only here CE will be occurred.

```
A(double d)
{
    S.O.P("A-double-param Con");
}
}
```

Status:- Compilation Error.

→ If we want to refer current class constructor by using "this" keyword then the respective "this" statement must be provided in another current class constructor, not from normal Java method, if we violate this rule then compiler will rise an error.

In the above two cases compiler will rise "call to this must be first statement in the constructor."

Ex:-

```
class A
{
    A()
    {
        S.O.P("A-con");
    }

    A(int i)
    {
        S.O.P("A-int-param-con");
    }

    void m1()
    {
        this(10); → It must be first statement
        S.O.P("m1-A");
    }
}
```

Class Test

```
PSum (String [args])
```

```
{  
A a = new A();  
a.m1();  
}  
}
```

Note: In this appn, "this" must be the first statement of the constructor.

States:- Compilation error → Call to 'this' must be first statement by the constructor. `this(10);`
↑ error.

- ⇒ Is it possible to refer more than one current class constructor by using "this" keyword from a single current class constructor?
- ⇒ No, Not at all possible, why because, to refer current class constructor by using "this" keyword then the respective "this" statement must be first statement. If we want to provide access more than one current class constructor by using "this" keyword then we have to provide more than one "this" statement within a single constructor, where only one 'this' statement i.e. first statement and all remaining "this" statement are not first statement.

```
Ex/  
Class A  
{  
A()  
}
```

`this(10);` `this(22.22f);` here, it can not be choose which is the first statement, and it can be collapsed for being the first statement means here CE will be S.O.P in ("A-Copy"), occurred.

```

A (int e)
{
    System.out("A-int-param-con");
}

A (float f)
{
    System.out("A-float-param-con");
}

```

Class Test

```

{
    PSVM (String [args])
    {
        A a = new A ();
    }
}

```

Status:- Compilation error.

③ To refer current class method:-

If we want to refer current class method by using "this" statement then we have to use the following Syntax:-

this.method-name ([param-list]);

Note:- To access current class method, it's not mandatory to use "this" keyword, we can access current class method without using any reference variable.

Eg:-

```

class A {
    void m1() {
        System.out.println("M1-A");
        m2();
        System.out.println("M1-A");
    }
    void m2() {
        System.out.println("M2-A");
    }
}

class Test {
    public static void main(String[] args) {
        A a = new A();
        a.m1();
    }
}

```

*m1(); method for jump on void m2();
 m2(); it will be possible that
 this.m2(); but when we use this.m2();
 M1-A, but when we use void m2(); and the printed
 on void m2(); and the printed*

*"M1-A"
 "M2-A"*

*We are calling
 this.m2(); if we not used
 output is only
 if can jump
 output is*

④ To return current class Object :-

If we want to return current class object by using "this"
keyword, then we have to use the following syntax.

return this;

=

P.T.O.

Class A

```
{  
    A getRef1()  
}
```

```
{  
    A obj = new A();  
    return obj;  
}
```

→ Here, the getRef1() value will be varied because, return obj will call new object for all the times which we want to print, so, that, it can be varied.

A getRef2()

```
{  
    return this;  
}
```

→ It's all three values are same because it will print by return this statement here, return this; is help to print same value for all the three times, because it can return same value by all the no. of times when we printed a.getRef2().

(Note:- suppose, if we print a.getRef1() 5 times, then it will be print same value on 5 times.)

Class Test

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

```
{  
    A a = new A();
```

```
S.0.P1n(a);  
S.0.P1n();  
S.0.P1n(a.getRef1());  
S.0.P1n(a.getRef2());  
S.0.P1n(a.getRef1());  
S.0.P1n();
```

```
S.0.P1n(a.getRef2());  
S.0.P1n(a.getRef2());  
S.0.P1n(a.getRef2());
```

OP:- {
 A@45a877
 @1372a1a
 @ad3ba9}

It can't be same with the ~~first~~ second time execution it can be change.

{
 @360be0
 360be0
 360be0}

In the above application, for every `getRef1()` method call, JVM will encounter 'new' keyword and JVM will create 'new' object every time, it will not provide Object Reusability.

→ In the above application, for every `getRef2()` method call, JVM encounter "return this" statement, JVM will return same object on which `getRef2()` method is called, this approach will improve Objects Reusability.

"Static" Keyword:-

→ Static keyword is able to improve sharability in java applications.

→ In java applications we are able to utilize static keyword in the following four ways.

- ① Static Variables ✓
- ② Static Methods ✓
- ③ Static Blocks ✓
- ④ Static import ✓

① Static Variables:-

If it's a java variable, it will be recognized and initialized at the time of loading the respective class by tcode.

- Static variables last modified value is shared to previous objects and to the future object which we are going to prepare.
- In java applications, static variables data will be stored in 'Method Area'.

→ In java applications, static variables are accessed either by using reference variable or by using class name.

Note:- To access static variables, always it is suggested to use class name directly.

→ In java applications, static variables must be declared at class level not as local variables.

Ex:-

Class A

{

 Static int i=10; → Here it is declared in class level.

 int j=0; → and here no compilation error will be occurred.

 Void m1() → Non-static method

{

 // Static int k=30; → Here compilation error occurred, because it is local variable and static

 S.O.P("m1-A"); → Compilation Error variables must be declared as class level not as local level

}

}

Class Test

{

 Psvm (String args)

 {

 S.O.P(a.i); → Note:- There all are int values hence, so we can't take it in " "(String) we can put there all as it is which is mentioned].

 A a=new A();

 S.O.P(a.i)

 A a1=null;

 S.O.P(a1.i);

 } // S.O.P(a1.i) → NullPointerException

31/08/2015 22

Con - static

Note:- In java applications, if we access any instance variable by using reference variable having "null" value then JVM will rise an exception like "java.lang.NullPointerException". If we want to access static variable by using reference variable having "null" value then JVM will not rise any exception, JVM will access the value of static variable.

Ex:-

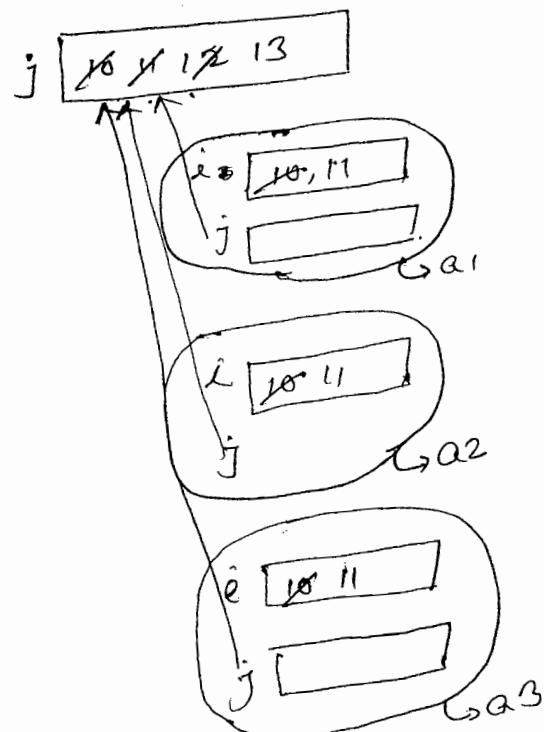
```

Class A{
    int i=10;
    static int j=10;
}

class Test{
    public static void main(String [] args){
        A a1 = new A();
        System.out.println("A.i=" + a1.i);
        a1.i=a1.i+1;
        a1.j=a1.j+1;
        System.out.println("A.i=" + a1.i + " " + a1.j); → 11 11
    }
}
  
```

```

A a2 = new A();
System.out.println("A2.i=" + a2.i);
a2.i=a2.i+1; → 10
a2.j=a2.j+1; 11
System.out.println("A2.i=" + a2.i + " " + a2.j); → 11 12
System.out.println("A2.i=" + a2.i + " " + a2.j); → 11 12
  
```



A Q3 = new A();

S.O.P17 (Q3.i + " + Q3.j); → 10 12

Q3.i = Q3.i + 1;

Q3.j = Q3.j + 1;

S.O.P17 (Q1.i + " + Q1.j); → 11 13

S.O.P17 (Q2.i + " + Q2.j); → 11 13

S.O.P17 (Q3.i + " + Q3.j); → 11 13

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Opp. CDAC, Balkampet Bakery,
Ameerpet, Hyderabad.

{
}
≡

Note:- In JAVA/J2EE applications for a particular class, bytecode will be loaded only one time but we can create any no. of objects.

Static Methods:-

Static method is a java method, it will be recognized and executed the moment when we access that method.

Note:- In java applications, without calling methods, methods won't be executed.

→ Static methods are accessed either by using class name directly or by using reference variable.

Note:- To access static methods, if we use reference variable then reference variable may or may not have

reference value. In java applications, it's possible to access static methods by using reference variables with null value. If we access instance methods by using a reference variable having null value then JVM will rise an exception like `java.lang.NullPointerException`.

→ Static methods are able to allow static members of the respective class directly, static methods are not allowing non-static members of the respective class directly.

Note:- If we access non-static members inside static methods then we must create object for the respective class and we must use the generated reference variable.

→ Static methods are not allowing "this" keyword in their body but to access current class static methods from some other methods then we are able to use "this" keyword.

Ex:-

Class A

```
{
    static int i=10; → static variable
    int j=20; → Non-static variable
    static void m1() → static method
}
```

S.O.P("m1 - A").

S.O.P(i), → Non-static keyword

// S.O.P(m1(this.i)); → Error

// S.O.P(j) → Error

Non-static members which is not allowed in static methods [if we want to use non-static members in static method then we have to refer above note]

Non-static members

reference variable

$a = \text{new } A();$ → Object created for allow Non-static member in Static method. With the help of create object we can allow Non-static member in static method.

}

Void m2()

{

→ reference variable with Non-static members it will be printed because of creating object.

Non-Static method

S.O.Pn("m2-A"),

{

this.m1(); → This is Non-static members and the non-static method will able to access the non-static members.

Two examples of calling static method by "A class name" and reference variable of "object A".

Class Test → Main class

PSVm(String[] args)

{

① A.m1(); → Static method call by Class Name A, because it can call by both object and class name.

A a1 = new A(); → Here, we take the reference variable a1 in object

② a1.m1(); → because the name 'a' is already used in above creation of object.

A a1 = new A(); → Here, static method call by object, because static method can call by both object and class name.

A a2 = null; → Here, m2() method is Non-static, so it will call by object only, never by class name.

A a2 = null; → Here, we create the object with 'null' value

and it can perform with static method but not with non-static.

// a2.m2(); → It can be perform and there no any exception will occurred

{

Non-Static method m2(), can not perform with this

object of Null value, so here, NullPointerException will occur.

≡

Q: Is it possible to display line of text on Command prompt, without using main() method?

⇒ Yes, it is possible to display a line of text on command prompt without using main() method, but by using static variable and static method.

Ex:- D:\java\am\|Test.java

Class Test

```

Static int m1(); → Static Variable
Static int m1() → Static method
{
    System.out.println("Welcome to Durga Software Solutions");
    System.exit(0); // To terminate program here itself.
    return 10; → return 10 will return the integer value to m1().
}

```

→ When we execute the above program, JVM will load Test class bytecode to the memory, at the time of loading main class bytecode to the memory, JVM will recognize and execute static variable, with this, static method. By execution of static method JVM will display the line of text on command prompt. After loading main class bytecode to the memory, JVM will check whether main() method is available or not, if not JVM will rise an exception like "java.lang.NoSuchMethodError: main".

In the above context, to avoid the exception message we have to terminate the program immediately after displaying the required text message on command prompt, for this, we have to use `System.exit(0);` method.

Note:- The above question and answer are valid upto Java 8 version, they are not valid in Java 9 and above versions, because in Java 9, JVM will not load main class bytecode without `main()` method.

If we run the above program in Java 9 version then we are able to get the following error message.

Error:- Main method not found in class Test, please define the main method as:

public static void main (String[] args).

(3) Static Blocks:-

- Static block is a set of instructions, which are recognized and executed at the time of loading the respective class bytecode.
- Static blocks are able to allow static members of the current class directly.
- Static blocks are not allowing non-static members of the current class directly, where if we want to access non-static members of the current class in static block then we have to create object for the current class and we have to use the generated reference variable.
- Static blocks are not allowing "this" keyword.

Class A

```

    {
        Static int i=10;           → static variable
        int j=20;                → non-static variable
    }

    Static {                      → static block
        S.O.Pn ("SB-A");
        S.O.Pn (i);
        //S.O.Pn (this.i) → error } → Here 'this' keyword is not allowed because it is non-static member
        //S.O.Pn (j) → error } → Here 'j' is not allowed because it is a non-static variable
                                → If we want to access int j in static block then we must create an object for access non-static members in static block. So, here object is created with reference variable a1 and with the help of this we can access the non-static members.
    }

    a1=new A();
    S.O.Pn (a1.j);
}

```

02/09/2015

```
Class Test
{
    public static void main(String[] args)
    {
        A a = new A();
    }
}
```

Q. Is it possible to display line of text on command prompt without using main() method, Static Variable ~~and~~ Static method.
~~Static block~~

⇒ Yes, it is possible to display a line of text on command prompt without using main() method, Static Variable, Static method, ~~and~~ but by using Static Block.

```
Class Test
{
    static {
        System.out.println("Welcome to Durga Software Solutions");
        System.exit(0);
    }
}
```

The above question and ans. is valid upto java8 version,
They are invalid in java7 version.

If we run the above program in java7 version then we are able to get the following error message.

Error:- main() method not found in class Test please define the

main method as:

public static void main (String [args])

Q. Is it possible to display a line of text on command prompt without using main() method, static variable, static method and static block?

→ Yes, it is possible to display a line of text on command prompt without using main() method, static variable, static method, and static block, but, by using static anonymous inner class of object class.

```
Class Test
{
    Static Object O = new Object
    {
        {
            S.O.P("Welcome to Durga Software Solutions");
            System.exit(0);
        }
    }
}
```

Note:- The above question and answer are valid upto java⁶ version & not valid in java⁷ version.]

If we run the above program in java⁷ version then we are able to get the following error message -

Error:- Main method not found in class Test, Please define the main method as:

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

④ Static import:-

In java applications, "import" statement can be used to make available classes and interfaces of a particular package in a present java file.

Similarly, to make available static members of a particular class or interface to the present java file we have to use "Static import".

If we make available static members of a particular class or interface to the present java file then we are able to access that static members without using class name and reference variable of the respective class or interface.

Syntax:-

① `import static Package-Name.Class-Name.*;`

→ It able to import all the static members of the specified class or interface.

② `import static Package-Name.Class-Name.Member-Name;`

→ It able to import only the specified member from the specified class or interface.

→

```
import static java.lang.Thread.*;
import static java.lang.System.out;
class Test
```

PSVM (String []args)

{
 Out.println (MIN_PRIORITY); → 1
 Out.println (NORM_PRIORITY); → 5
 Out.println (MAX_PRIORITY); → 10
}

Note:- Static import feature is provided by JAVA along with its JDK5.0 version.

Static context/Static Flow of Execution:-

In java applications, static context is represented by the following three elements.

- ① Static Variable
- ② Static Method
- ③ Static Block

Where static variables and static blocks are executed automatically at the time of loading the respective class bytecode to the memory. but static methods are executed the moment when we call static method.

```

Class A
{
    static → static block.
    {
        S.O.P("SB-A");
    }
    static (int) m1() → static method
    {
        S.O.P("M1-A");
        return 10; → It will return the integer value for m1() method only.
    }
    static int i = m1(); → static variable (it is not required to use static variable
    {                                         but here it is compulsory to use static variable)
}

```

Here, int is available in the class file and can be called by reference value as it is mentioned, then we must give the reference variable for static method gives return 10 for static int m1().

```

Class Test
{
    PSVM (String [] args)
    {
        A a = new A();
    }
}

```

Ex:-

Class A

{

 Static int m1(); → static method

 S.O.P("m1-A");

 return 10; → It will return the integer value for m1() method.

 Static

 { → static block

 S.O.P("SB-A");

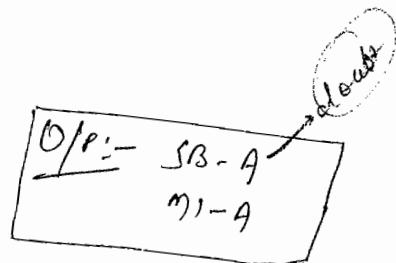
 Static int i=m1(); → static variable

Class Test

{ Dsum(String args)

 A a1=new A();

 A a2=new A();



Ex:-

Class A

{

 A() → Constructor

 S.O.P("A-Cong");

 Static int i=m1(); → static variable

 S.O.P("IB-A");

 int m2(); → Non-static method

 S.O.P("m2-A");

03/09/2015 45

return 30; → It will return the integer value for non-static (int m2()) method.

} static → static block

{ S.O.P("SB-A");

int j=m2(); → Non-static variable
Static int m1() → static method

{ S.O.P("m1-A"); → It will return the integer value for static int m1() method

Class Test{
 return 30; → It will return the integer value for static int m1() method
}
 {
 PSVM (String []args)
 {
 A a1=new A();
 A a2=new A();
 }
 }
}

O/P:- M1 - A
SB - A
IB - A
M2 - A
A - con
IB - A
M2 - A
A - con

Class.forName():-

Consider the following program.

```
Class A {  
    Static { → static block  
        System.out.println ("Class loading"); -①  
    }  
    A () → constructor  
}
```

```
System.out.println ("Object creating"); -②  
}
```

```
Class Test {  
    PSVM (String []args) {  
        A a=new A();  
    }  
}
```

Class.forName()

→ Static block executes
on class loading

[Static blocks executes
or runs at any time
but can't make any
functionality in it]

→ In the above application, if we access a class constructor by using "new" keyword then JVM will perform the following two actions automatically.

- ① Class Loading ~
- ② Object Creating. ~

As per the application requirement, if we want to load class bytecode to the memory only without creating object then we have to use the following method from "java.lang.Class" class.

→ Public static Class.forName(String class_name) throws ClassNotFoundException - IOException.

Ex:- Class C = Class.forName("A"); → it will load the runtime Class it will work like JVM because for ex:- if we write a program in jre and save it and after that when we use Class.forName("A") the "A" class programme in jre will copied in the present programme if we not use class A in the programme

When JVM encounters the above instruction, JVM will perform the following actions.

- ① JVM will take the specified class name from forName() method.
- ② JVM will search for the specified class .class file at current location at java predefined library and at the locations referred by "Classpath" environment variable.
- ③ If the required .class file is not identified at all the above specified locations then JVM will give an exception like "java.lang.ClassNotFoundException".
- ④ If the required .class file is available at either of the above locations then JVM will load its bytecode to the memory.

- ⑤ At the time of loading class bytecode to the memory static code will be executed.
- ⑥ After loading class bytecode to the memory, JVM will collect the respective loaded class metadata and JVM will store that metadata in the form of "java.lang.Class" object.

Note:- Class metadata includes the details like name of the class, super class details, implemented interfaces details, variables details, method details, constructor details ..

In the above context, if we want to create object for the loaded class then we have to use the following method from java.lang.Class.

```
public Object newInstance() throws
InstantiationException, IllegalAccessException
```

Ex:- Object obj = C.newInstance();

When JVM executes the above instruction, JVM will perform the following actions.

- ① JVM will go to loaded class metadata available in java.lang.Class object and search for a 0-arg constructor and go - Private constructor.
- ② If the required constructor is available then JVM will create object by executing the constructor.
- ③ If 0-arg constructor is not available in the loaded class then JVM will raise an exception like "java.lang.InstantiationException".

④ If non-private constructor is not available then JVM will rise an exception like

"java.lang.IllegalAccessException"

⑤ If no constructor is both parameterized and private then JVM will rise

"java.lang.InstantiationException"

~~for - int~~
~~Class A~~

{
 Static → static block
 }
 }

S.O.P("Class Loading");

AC → Constructor with 0-arg/non-private Constructor

{
 S.O.P("Object Creating");
}

}

Class Test

{
 PSVM (String [] args) throws Exception

 Class c = Class.forName ("A");

 Object obj = c.newInstance ();

}

}

2

It is used for loading the specified class
eg class is not defined in present APPN and
this class is defined in jre the Class.forName
method can take it from jre and copied
in present application as it is. Class.forName
is worked like JVM to load the defined
and predefined class.

It is used to install the constructor, if object is
not created, then we have to C.getInstance() for
installing here constructor.

Example 1:-

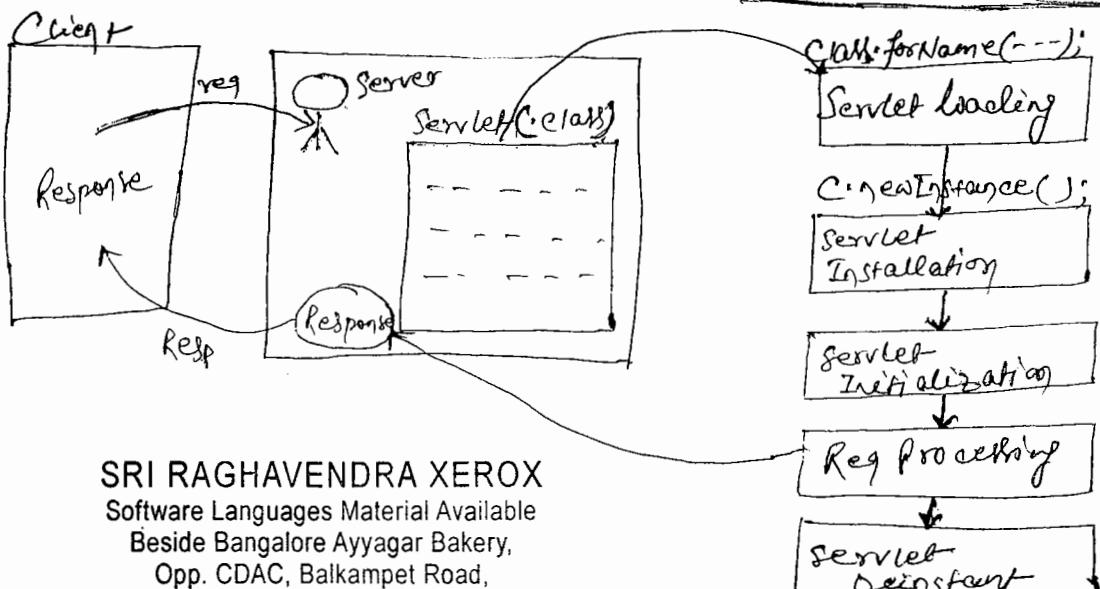
In JDBC, Driver is a class, it can be used to map java representation to database representations and database representations to java representations. To execute JDBC applications, it is mandatory to load driver class bytecode to the memory without creating object. To achieve this requirement we have to use `Class.forName(-)` method.

```
Class.forName("sun.jdbc.odbc.jdbcodbcDriver");
Class.forName("Oracle.jdbc.OracleDriver");
```

(Note: Here, we have to load driver class bytecode to the memory, without creating object. here, driver class is a predefined class in `java.sql` and by the use of `Class.forName` we can access it directly, without using object.)

Example 2:- In java, Only J2EE applications are executed by using main() method. All server side components in J2EE are executed by following their own lifecycle actions or lifecycle methods.

In All the server side components lifecycle actions "Loading" and "Instantiation" are two common lifecycle actions servers will use "`Class.forName(-)`" method, "`C.newInstance()`" method;



Factory Method:-

Factory method is a java method, it has to return the same class object / different class objects also depending on our application requirements.

Factory method is an idea provided by "factory method design pattern".

```
Class A
{
    private A() { } // private constructor
    {
        System.out.println("A-con");
    }
    void m1()
    {
        System.out.println("m1-A");
    }
    static A getRef() { } // Here, by the getRef() method,
    // we can create an object in the
    // same class.
    {
        A a = new A();
        return a; // with getRef() intRef() ... we must have to
        // use "return" type. but with voidRef()
        // we can't use the return method.
    }
}

class Test
{
    public static void main(String[] args)
    {
        A a = A.getRef();
        a.m1();
    }
}
```

In java, there are two types of factory methods

- ① Static factory method
- ② Instance factory method.

① Static factory method:-

Static factory method is a static method return the same class object.

Ex:-

```
Class c = Class.forName("...");
```

```
NumberFormat nf = NumberFormat.getInstance();
```

```
DateFormat df = DateFormat.getDateInstance(..., ...);
```

```
ResourceBundle rb = ResourceBundle.getBundle(...);
```

② Instance factory method:-

Instance factory method is an instance method return the same class object.

Ex:- Majority of String class methods are instance factory methods.

```
String str = new String("abc");
```

```
String str1 = str.trim();
```

```
String str2 = str.concat("def");
```

```
String str3 = str.toLowerCase();
```

* Singleton Class:-

If any java class allow to create only one object then that java class is called as singleton class.

Singleton class is an idea provided "Singleton design pattern".

If we want to implement singleton class in java apps then we

have to use the following steps.

- ① Declare a class with private constructor.
 - ② Provide a static factory method with the following implementation.
 - a) Check whether any object is created or not for the current class. If any object is existed/created previously then return the same object reference without creating new object.
 - b) If no object is existed previously then create new object and return the generated reference value.
- ~~To implement the above logic in factory method, we have to use the following steps.~~
- ① Declare a static reference variable of the same class with null value.
 - ② In factory method, check static reference variable value is equals to "null" or not by using '==' operator.
 - ③ If static reference variable value is null then create object for the current class and return the generated reference variable.
 - ④ If static reference variable value is not null then return the same reference variable.

~~Ex:-~~ Class A

```
{  
    static A a=null; //a=abc123  
    private A() {  
    }  
}
```

```

Static A getRef()
{
    if(a==null)
    {
        a=new A();
    }
    return a;
}
}

Class Test
{
    PSVM (String[] args)
    {
        S.O.Pln (A.getRef()); // A@abc123
        S.O.Pln (A.getRef()); // A@abc123
        S.O.Pln (A.getRef()); // A@abc123
    }
}

```

Alternative logic for Singleton Class:-

```

Class A
{
    Static A a=null; // a=abc123
    $tatic
    {
        a=new A();
    }
    private A()
    {
    }
}

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

```
Static A getRef()
```

```
}
```

```
return a;
```

```
{
```

```
}
```

```
Class Test {
```

```
PSVM (String [] args) {
```

```
S.O.P1N (A.getRef());
```

```
S.O.P1N (A.getRef());
```

```
S.O.P1N (A.getRef());
```

```
}
```

~~Another alternative for Singleton Class:-~~

```
Class A
```

```
{
```

```
Static A a = new A();
```

```
private A()
```

```
{
```

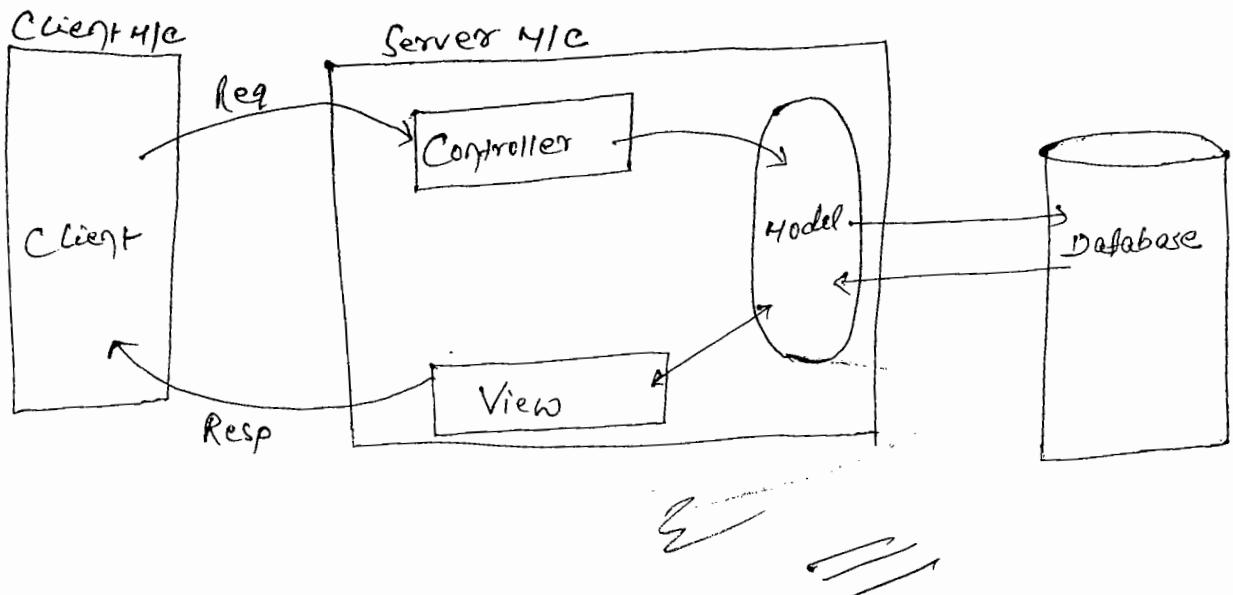
In MVC based application, a class must be used as controller and as per MVC rules and regulations controller must be single that is controller class, must allow to create only one object, so that controller class must be Singleton class.

Ex:-

Struts is MVC based framework, if able to use ActionServlet as a controller, it must allow to create only one object, so, that ActionServlet must be a Singleton class.

Ex:-

JSF is MVC based framework, if able to use FacesServlet as controller, it must allow to create only one object, so, that FacesServlet must be a Singleton class.



Final Keyword:-

'final' is a java keyword, it able to declare Constant expressions.

→ In java application, We are able to utilize final keyword in the following three ways.

- ① final Variables ✓
- ② final methods ✓
- ③ final Classes. ✓

① final variables:-

final variable is a java variable, it will not allow modifications of its values.

→ final variables are not allowing re-assignments.

Example:-

In bank applications, when we create an account and where if account details like account holder name, account holder address details, ... are wrong then bank manager is able to modify the above specified account data, where A/c no. is not possible to modify. In this context to achieve the requirement like a/c no. and its values are not possible to modify we have to declare 'accNo' variable as "final" variable.

② final methods:-

final method is a normal java method, it will not allow method overriding.

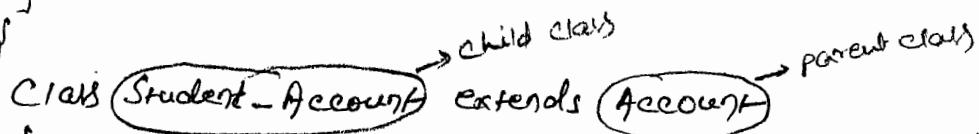
Ex:- In any bank application, in any account if we want to fix minimum balance value is 5000 irrespective of the account

which we are creating then the method `getMinBal()` must be declared as `final`.

Class Account

```
final int getMinBal() {  
    return 5000;  
}
```

`getMinBal()` method with `int` not with `void`
so we can return the value `int getMinBal()`,
if here `void getMinBal()` method is available,
then we cannot return any value, because `void`
haven't any return type.



```
int getMinBal() {  
    return 10000;  
}
```

Status:- Compilation error. (doubt)

→ here the `int getMinBal()` is overwriting because of above `Account` class is declared as `final`. Here we cannot change or write again, because it can't be changed. So, that here is compilation errors occurred.
→ If the above `Account` class not declared as `final` and `Student-Account` class is declared then, it stop after `Student-Account` class, and it will print both 5000 and 10000 but 10000 is `final`, then we can't put any other integer after declared `final`, we can put any integer before declaration of `final` not after.

final class:-

final class is a normal java class, it will not allow inheritance.

→ In java applications, super class never be declared as final class.

final class Employee

```
Class Permanent-Employee extends Employee
```

Status:- Compilation error.

Note:- In java applications, only super class are not possible to declare final classes but it is possible to declare sub classes are final classes.]

In java applications, to declare Constant Variables JAVA has provided a separate convention like "public static final".

Where the purpose of "public" is to make available constant variables throughout our java application.

Where the purpose of declaring constant variables as "static" is to access constant variables by using class name directly, not by creating reference variable.

Where the purpose of declaring constant variables as "final" is to fix a particular value to the variable throughout the application.

Class User

{

 public static final int MIN-AGE=18;

 public static final int MAX-AGE=25;

}

Class Test {

 public static void main (String[] args)

{

 System.out.println (User.MIN-AGE);

 System.out.println (User.MAX-AGE);

}

}

=

Ex:-
Class User_Status

```

{
public static final String AVAILABLE = "User is Available";
public static final String BUSY = "User is Busy";
public static final String IDLE = "User is Idle";
}
  
```

Class Test

```

{
public void main (String [] args)
{
System.out.println (User_Status.AVAILABLE);
System.out.println (User_Status.BUSY);
System.out.println (User_Status.IDLE);
}
}
  
```

In java applications, if we use the above approach to declare constant variables, then we are able to get the following problems:-

- ① Every time declaration of constant variables we have to provide "public static final" explicitly.
- ② This approach is able to allow diff. data types to represent one thing, it will provide type losing feature in java app?
- ③ In this approach, When we access constant variables then we are able to get values of the constant variables, where constant variable values may reflect or may not reflect the actual meaning of the constant variables.

To overcome all the problems we have to use "enum".

In case of "enum":-

- ① All the constant variables are by default "public static final", not required to declare explicitly.
- ② All the constant variables are by default the same enum type, not required to provide any other data type, it will improve type safety in java applications.
- ③ All the constant variables are "named constants" by default they will display names of the constant variables, not values.

Syntax:-

[Access-Modifier] enum Enum-Name
{
 --- List of Constant Variables ---
}

Ex:-

```
enum User_Status  
{  
    AVAILABLE, Busy, IDLE;  
}  
class Test  
{  
    main (String [] args)  
    {  
        System.out.println (User_Status.AVAILABLE);  
        System.out.println (User_Status.Busy);  
        System.out.println (User_Status.IDLE);  
    }  
}
```

If we compile the above User_Status enum then compiler will translate "User_Status" enum into the following class.

final class User_Status extends java.lang.Enum

```

{
    public static final User_Status AVAILABLE;
    public static final User_Status BUSY;
    public static final User_Status IDLE IDLE;
    ...
}
```

In java by default enum is final class, it will not allow inheritance.

In java all the enums are sub types to java.lang.Enum class.

* It is an example of next page (103) topic.

enum Apple *User Defined*

```

{
    A(500), B(250), C(100);
    int price;
    Apple(int price)
    {
        this.Price = price;
    }
    public int getPrice()
    {
        return price;
    }
}
```

Internal translated Coding part by Compiler

```

final class Apple extends java.lang.Enum
{
    public static final Apple A = new Apple(500);
    public static final Apple B = new Apple(250);
    public static final Apple C = new Apple(100);
    int price;
    Apple(int price)
    {
        this.Price = price;
    }
    public int getPrice()
    {
        return price;
    }
}
```

In java appⁿ, we are able to utilize enum like a class
and we are able to declare normal variables, normal methods,
constructors, ... along with constant variables along with inside
the enum, but enums are utilized mainly for declaring Constant
variables.

```
enum Apple
{
    A(500), B(250), C(100);
    int price;
    Apple(int price)
    {
        this.price=price;
    }
    public int getPrice()
    {
        return price;
    }
}

class Test
{
    public static void main(String[] args)
    {
        System.out.println("A-Grade Apple:" + Apple.A.getPrice());
        System.out.println("B-Grade Apple:" + Apple.B.getPrice());
        System.out.println("C-Grade Apple:" + Apple.C.getPrice());
    }
}
```

If we compile the above program then compiler will translate Apple gun into the following class. (See on previous page for example) *page 102*

Importance of main() method in java:-

The main intention to write main() method in java applications is,

- (1) To manage appn logic in java application.
- (2) To define starting point and ending point for the application execution.

[Note:- In java apps, main() method starting point is the starting point of the application execution and main() method ending point is the ending point of the application execution.]

Syntax:-

```
public static void main(String[] args)
{
    --- Application logic. ---
}
```

[Note:- main() method is not predefined method, and it is not user defined method, it is a conventional method with predefined method prototype and with user defined implementation part.]

JVM was prepared in such a way that to access this convention method with the fixed prototype in order to execute user defined application logic available inside main() method.

Q/ What is the requirement to declare main() method as public?

- ⇒ In java applications, to execute application logic JVM must access main() method. To access main() method by JVM, main() method scope must be available to JVM.
- If we declare main() method as private then main() method scope is not available to JVM, so, that JVM is unable to access main() method, because private main() method is available upto the main class.
- If we declare main() method as "default" then main() method is available upto the present package, it is not available to JVM, so that JVM is unable to access main() method.
- If we declare main() method as "protected" then main() method is available upto the present package and sub classes available in other packages, but not to the JVM, because JVM is available inside java software, not in the present package and JVM class is not child class to our main class, so that JVM is unable to access main() method.
- If we declare main() method as "public" then main() method is available throughout the application and throughout the system, main() method scope is available to JVM, so that, JVM is able to access main() method.
- The main intention to declare main() method as "public" is to bring main() method scope to JVM.

Note:- In java applications, if we declare main() method without public then compiler will not rise any error but JVM will provide the following.

JAVA6: Main method not public

JAVA7: Error: Main method not found in class Test, please define main method as:
public static void main(String[] args)

Q: What is the requirement to declare main() method as "Static"?

⇒ java creators have designed JVM in such a way that to access main() method by using class name directly. As per the JVM predefined implementation part, we must declare main() method as static method, because in java programming language only static methods are eligible to access by using class name directly.

In the above case, JVM will get main class name from command prompt which we specified along with "java" command on command prompt in order to access main() method.

Note:- Java creators have prepared JVM to access main() method directly by using class name without creating object in order to make JVM as light weight component.]

Note:- If we declare main() method without static keyword in java applications, then compiler will not rise any error but JVM will provide the following.

JAVA6: java.lang.NoSuchMethodError: main

JAVA7: Main method is not static in class Test, please define the

main method as:

public static void main(String[] args).

Ex:-

Q. Is it possible to interchange public and static in main() method Prototype (SYNTAX)?

→ Yes, it is possible to interchange both public and static in main() method syntax, because, in java applications normal java methods are able to allow more than one access modifiers in any order.

Ex:-

Class Test {

 static public void main(String[] args)

 }

}

Q. What is the requirement to provide "void" return type to main() method?

→ In java applications main() method starting point is the starting point for application execution and main() method ending point is the ending point for application execution.

→ In java applications, to execute application logic, JVM will create "Main Thread" at the starting point of the main() method and JVM will terminate "Main Thread" at the

ending point of main() method as per the above java Convention.

To preserve the above java Conventions, we must terminate application logic at the ending point of the main() method, for this, we must not return any value from main() method, so, that we must provide 'void' as return type to main() method.

Note:- If we provide(declare) main() method without "void" return type in java applications, then compiler will not give any error but jvm will provide the following.

JAVA6: java.lang.NoSuchMethodError: main

JAVA7: Error: Main method must return a value of type "void" in class Test, Please define the main method as :

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

Note:- The main intention to give name for this Conventional method like "main" is to show up its importance in java Application.

11-09-2015

Q. What is the requirement to provide parameters to main() method?

⇒ In java applications, we are able to provide input data to the following three ways.

- ① Static Input
- ② Dynamic Input
- ③ Command Line Input/Arguments.

① Static Input:-

If we provide input data to java programs at compile time

Writing java programs such that input data is called by

Ex:-

Class A

{

int i=10; → Static input

int j=20; → Static input

~~int k=i+j;~~

Void add()

{

int k=i+j;

S.O.P(k);

}

Dynamic Input:-

If we provide data to the java programs at runtime then
that input data is called as

Ex:-

D:\apps>javac Add.java

D:\apps>java Add

Enter First Value : 10 → I/P.

Enter Second Value : 20 → I/P

Addition : 30 → O/P

Command Line Input/Arguments:-

If we provide Input data along with "java" command on Command prompt then that input data is called as Command Line Input or Command Line Arguments.

Ex:-

D:\apps>javac Add.java

D:\apps>java Add 10 20.

- If we provide command line input along with java command on command prompt then JVM will read all the command line input, JVM will store command line input in the form of `String[]` and JVM will pass the generated `String[]` to `main()` method at the time of calling `main()` method.
- Therefore, the purpose of `main()` method parameter is to store all the command line input in the form of `String[]` in order to make available them to the java program.

```
D:\javatam>javac Test.java
D:\javatam>java Add 10 20
```

`String[]`

"10" | "20"

Class Add

}

psvm(`String[] args`)

{

}

Q. What is the requirement to provide String data type as parameters to main() method?

- ⇒ In java applications, developer to developer the types of Command line input may be varied. Even though variables types of command line input is provided by the developers our main() method must store all the types of command line input.
- Due to the above reason, main() method must require a data type which store any type of data. In JAVA/J2EE technologies, Only string data type is able to store any type of data, therefore main() method must require string data type as parameters.

2

Q. What is the requirement to provide array type as parameter to main() method?

- ⇒ In java applications, from developer to developer number of Command line input may be varied. Even though we provide provide variable number of command line input our main() method must store all the command line input. In JAVA/J2EE technologies, Only array types are able to store more than one value, so that, main() method must require array type as parameter.

3

107

Ex:-

Class Test

{

public static void main(String[] args)

{ for (int i=0; i<args.length; i++)

{ System.out.println(args[i] + " "); }

}

I/P: 10 22.22f 34.34 true 'A' abc.

O/P: 10. 22.22f 34.34 true 'A' abc (~~gave~~)

Ex:- Class Test

{

PSVM (String[] args) throws Exception

{ int val1 = Integer.parseInt(args[0]);

int val2 = Integer.parseInt(args[1]);

S. O. P() ("Add : " + (val1 + val2));

S. O. P() ("Sub : " + (val1 - val2));

S. O. P() ("MUL : " + (val1 * val2));

}

Z

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

12-09-2015

Q Find the valid syntax of main() method from the following list!

- ① PSVM (String [Jargs) → v
- ② PSVM (String [Jabc) → v
- ③ PSVM (String args[]) → v
- ④ PSVM (String [Jargs) → v
- ⑤ PSVM (String[] args) → Invalid
- ⑥ PSVM (String () args) → Invalid
- ⑦ PSVM (String [] args) i → int → Invalid
- ⑧ PSVM (String [] args) f → final → Invalid (because static is missing)
no problem for final
- ⑨ PSVM (String [] args) s → static, f → final. → Invalid
- ⑩ Static public void main (String[] args) → valid
- ⑪ static void main (String[] args) → Invalid (public is missing)
- ⑫ public void main (String[] args) → Invalid (static is missing)
- ⑬ public static void main (int[] args) → Invalid (for int[]).
- ⑭ public static void main (String ... args) → Valid.

Q. Is it possible to provide ~~one~~ more than one main() method with in a single java application?

⇒ Yes, it is possible to provide more than one main() method within a single java appn, but in different classes, not in a single class.

In the above context, JVM will execute a main method whose class name which we specified along with "java" command on Command prompt,

Ex:- [File Name : D:\javatam\abc.java]

Class A

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

Class B

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

Command Prompt

= D:\javatam>javac abc.java

D:\>java A

[Main() - A]

D:\javatam>java B

[Main() - B]

In the above application it is possible to access one class main() method from another class main() method just like a normal static method and like String[] parameter that is by using class name and by passing String[] as parameter.

D:\javatam\abc.java

Class A

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

```

String[] str = {"AAA", "BBB", "CCC"};
B.main(str);
B.main(args);
}
}

Class B
{
public void main(String[] args)
{
System.out.println("main() - B");
}
}

```

Command prompt
javac -a b.java
java A
Output:-
main() - A
main() - B
main() - B

Q. Is it possible to overload main() method?

→ Yes, it is possible to overload main() method but it is not possible to override main() method in java applications, because in java applications static method overloading is possible, but static method overriding is not possible.

Note:- In the above case, even though we able to provide more than one main() method with different parameter list JVM will execute only one main() method which contains String[] parameter.]

Ex:- D:\java7am\Test.java

Code

Ex:-

14-09-2015

(109)

```

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

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

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

```

Command prompt:

D:\Java\javatut\javatest.java

java Test

O/P:- String[] - main()

Relationships in Java:-

- ① HAS-A Relationship
- ② IS-A Relationship
- ③ USES-A Relationship.

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

As part of java application development, there may be a chance to get some design problems while optimizing code and reducing execution time, to Overcome these design problems we have to define some relationships between entity classes.
 There are three types of Relationships between entity classes in java.

Q: What is the differences between HAS-A Relationship and IS-A Relationship?

- HAS-A Relationship is able to define associations between entity classes, which are able to improve data navigation between entity classes and communication between entity classes.
- IS-A Relationship is able to define inheritance relation b/w entity classes, it able to improve code reusability in java applications.

Associations in java:-

There are four types of Associations between entity classes in order to improve data navigation between entity classes and to improve communication between entity classes.

- ① One-To-One Association.
- ② One-To-Many Association.
- ③ Many-To-One Association.
- ④ Many-To-Many Association.

In java applications, to achieve associations we have to declare either single reference or array of references of an entity class in another entity class.

Ex:- Class Wheel

```
{  
    ...  
}
```

Class Car

```
{
```

Wheel[] wheels;

```
    Engine engine;
```

```
Class Engine
{
}
```

① One-To-One Association :-

It is a relation between entities, where one instance of an entity should be mapped with exactly one instance of another entity.

Ex:- Every Employee has exactly one account.



Ex:- Class Account

```

String accNo;
String accName;
String accType;
Account(String accNo, String accName, String accType)
{
    this.accNo = accNo;
    this.accName = accName;
    this.accType = accType;
}
  
```

parameterized constructor
lets us give all
the parameters
which is used in
Account class.

Class Employee

```

String eid;
String ename;
float esal;
String eaddr;
Account acc;
  
```

Employee (String eid, String ename, String eaddr, float esal, Account acc)

```
{  
    this.eid = eid;  
    this.ename = ename;  
    this.esal = esal;  
    this.eaddr = eaddr;  
    this.acc = acc;
```

```
}  
public void getEmpDetails()  
{  
    System.out.println ("Employee details");  
    System.out.println ("-----");  
    System.out.println ("Employee Id :" + eid);  
    System.out.println ("Employee Name :" + ename);  
    System.out.println ("Employee Salary :" + esal);  
    System.out.println ("Employee Address :" + eaddr);  
    System.out.println ("Account details");  
    System.out.println ("-----");  
    System.out.println ("Account No :" + acc.aceno);  
    System.out.println ("Account Name :" + acc.accname);  
    System.out.println ("Account Type :" + acc.acctype);  
}
```

Class OneTo OneEx

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

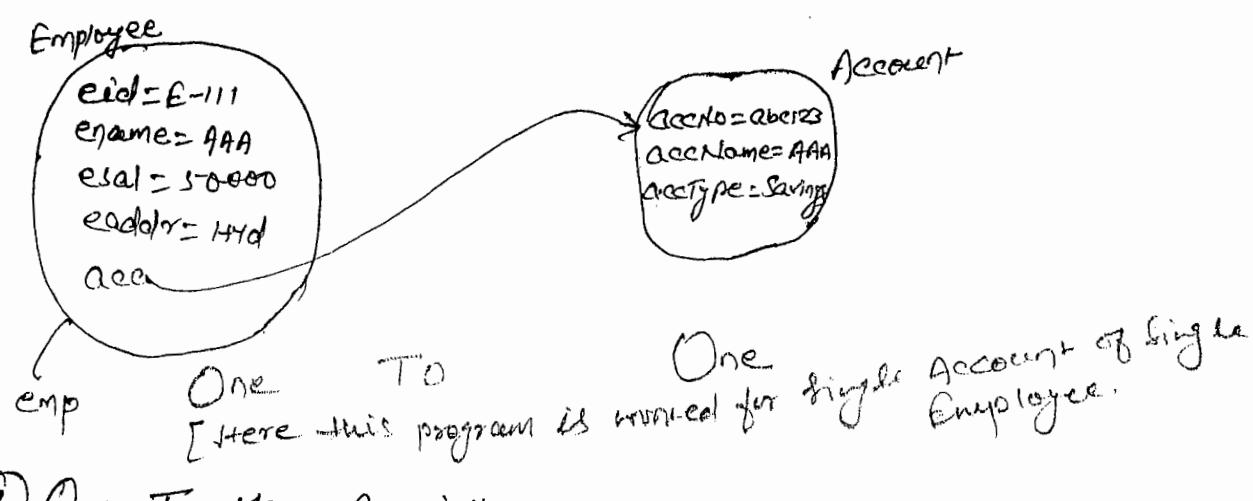
Object created for call the
Account class.

```
Account acc = new Account ("abc123", "AAA", "Savings");
```

```
Employee emp = new Employee ("E-111", "AAA", "50000", "Hd", acc);
```

```
emp.getEmpDetails();
```

Object created for call the
Employee class.
getEmpDetails
methods used for
call and print the details of Employee.



② One-To-Many Association:-

It is a relation between entities, where one instance of an entity should be mapped with multiple instances of another entity.

Ex:- Single Department has Multiple Employees.



Class Employee

```

    {
        String eid;
        String ename;
        String eaddr;
    }
    
```

→ Class level Variable
(Global Variable)

Employee(String eid, String ename, String eaddr)

```

    {
        this.eid = eid;
        this.ename = ename;
        this.eaddr = eaddr;
    }
    
```

Class Department

```

    {
        String did;
        String dname;
        Employee[] emps;
    }
    
```

Here [] is used for more than one/Many Association.

```

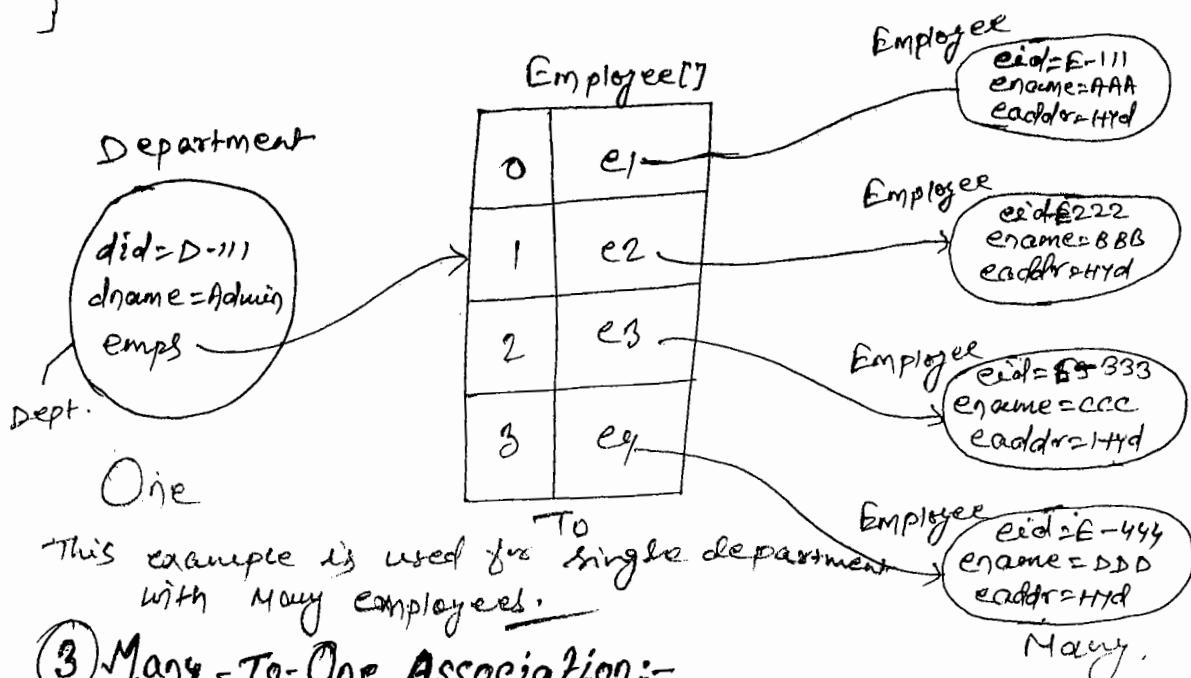
Department (String did, String dname, Employee [] emps) → parameterized
    {
        this.did = did;
        this.dname = dname;
        this.emps = emps;
    }
    public void getDeptDetails() → O-arg constructor for print the
    {                                         single Department details with true
        S.O.Pln("Department Details");      more no. of employee.
        S.O.Pln("Department Id :" + did);
        S.O.Pln("Department Name :" + dname);
        S.O.Pln();
        S.O.Pln("Eid |> Ename |> Eaddr");
        {
            S.O.Pln("-----");
            for (Employee e : emps)
            {
                S.O.Pln(e.eid + " |> " + e.ename + " |> " + e.eaddr);
            }
        }
    }
    class OneToMany
    {
        public static void main (String [] args)
        {
            Employee e1 = new Employee ("E-111", "AAA", "Hyd");
            Employee e2 = new Employee ("E-222", "BBB", "Hyd");
            Employee e3 = new Employee ("E-333", "CCC", "Hyd");
            Employee e4 = new Employee ("E-444", "DDD", "Hyd");

            Employee [] emps = new Employee [4]; → Here, object created {} or
            emps[0] = e1;                                define many no. of Employees.
            emps[1] = e2;
            emps[2] = e3;
            emps[3] = e4;
        }
    }

```

Department dept = new Department("D-111", "Admin", emps);

dept.getDeptDetails(); → from dept.getDeptDetails method we can call and print the department details.



(3) Many-To-One Association:-

It is a relation between ~~entity~~ entities, where multiple instances of an entity should be mapped with exactly one instance of another entity.

Ex:- Multiple Students have joined with Single Branch.



Ans:-

Class Branch

String bid;

String bname;

Branch (String bid, String bname)

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
{  
    this.bid = bid;  
    this.bname = bname;  
}
```

Class Student

```
{  
    String sid;  
    String sname;  
    String saddr;  
    Branch branch;  
    Student(String sid, String sname, String saddr, Branch branch)  
    {  
        this.sid = sid;  
        this.sname = sname;  
        this.saddr = saddr;  
        this.branch = branch;  
    }  
}
```

```
public void getEmpDetails()  
{  
    S.o.println("Student Details");  
    S.o.println("-----");  
    S.o.println("Student Id :" + sid);  
    S.o.println("Student Name: " + sname);  
    S.o.println("Student Address :" + saddr);  
    S.o.println("Branch Id :" + branch.bid);  
    S.o.println("Branch Name :" + branch.bname);  
    S.o.println();  
}
```

(11.8)

Class ManyToOneEx

PSVM (String[] args)

Branch branch = new Branch ("B-111", "CS");

Student std1 = new Student ("S-111", "AAA", "HYD", branch);

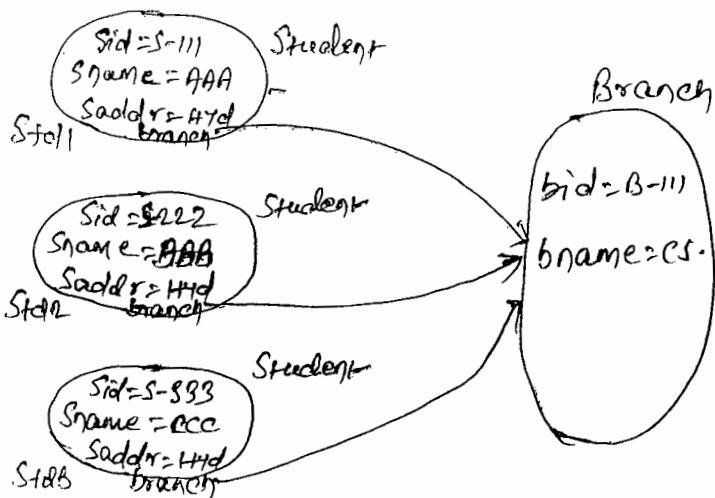
Student std2 = new Student ("S-222", "BBB", "HYD", branch);

Student std3 = new Student ("S-333", "CCC", "HYD", branch);

std1.getStudentDetails();

std2.getStudentDetails();

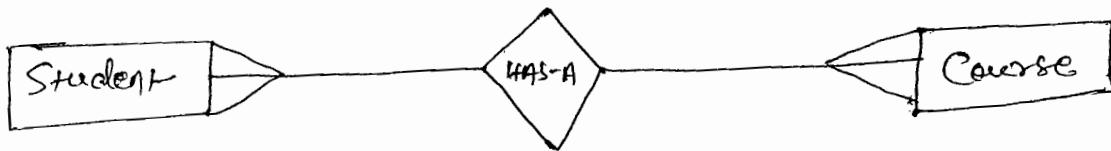
std3.getStudentDetails();



④ Many-To-Many Association:-

It is a relation between entities, where multiple instances of an entity should be mapped with multiple instances of another entity.

Ex:- Multiple Students have joined with multiple Courses.



Ex:-

Class Course:

```

{
    String cid;
    String cname;
    String ccost;
}
Course(String cid, String cname, int ccost)
{
    this.cid = cid;
    this.cname = cname;
    this.ccost = ccost;
}
  
```

Class Student

```

{
    String sid;
    String sname;
    String saddr;
    Course[] crs;
}
Student(String sid, String sname, String saddr, Course[] crs)
{
    this.sid = sid;
    this.sname = sname;
    this.saddr = saddr;
    this.crs = crs;
}
  
```

public void getStudentDetails()

(14)

```

S.O.Pln ("Student details");
S.O.Pln ("-----");
S.O.Pln ("Student Id :" + sid);
S.O.Pln ("");
S.O.Pln ("");
S.O.Pln ("CID\cename\ccost");
S.O.Pln ("-----");
for (Course c: crs)
{
    S.O.Pln (c.cid + "\t" + c.cname + "\t" + c.ccost);
}
S.O.Pln ();
}

```

Class ManyToManyEx

{ PSRM (String[] args) }

Course c1 = new Course ("C-111", "C", 1000);

Course c2 = new Course ("C-222", "C++", 2000);

Course c3 = new Course ("C-333", "JAVA", 5000);

Course[] crs = new Course[3];

crs[0] = c1;

crs[1] = c2;

crs[2] = c3;

Student std1 = new Student ("S-111", "AAA", "Hyd", crs);

Student std2 = new Student ("S-222", "BBB", "Hyd", crs);

Student std3 = new Student ("S-333", "CCC", "Hyd", crs);

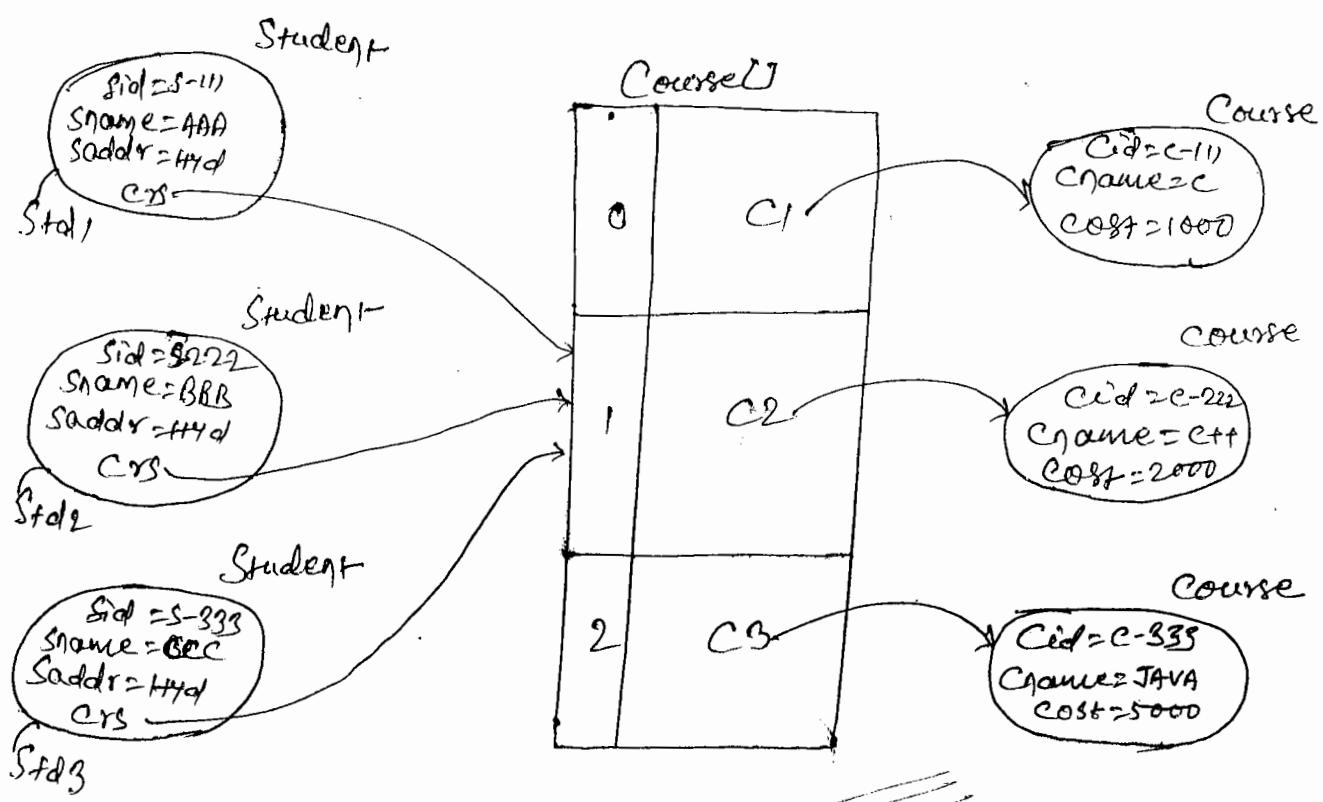
Std1.getStudentDetails();

Std2.getStudentDetails();

Std3.getStudentDetails();

}

z



Associations:-

In java applications, associations are available in the form of the following two ways.

- (1) Composition
- (2) Aggregation.

Q. What are diff. b/w composition and Aggregation?

→ (1) Composition is strong association between entities.

Aggregation is weak association between entities.

(2) In case of composition, if we destroy container object then there is no chance of existing contained object.

In case of Aggregation, if we destroy container object then there is a chance to exist contained object.

(3) In case of composition, life time of the contained object is totally depend on the life time of the container object.

In case of Aggregation, life time of the contained object is not depending on life time of the container object.

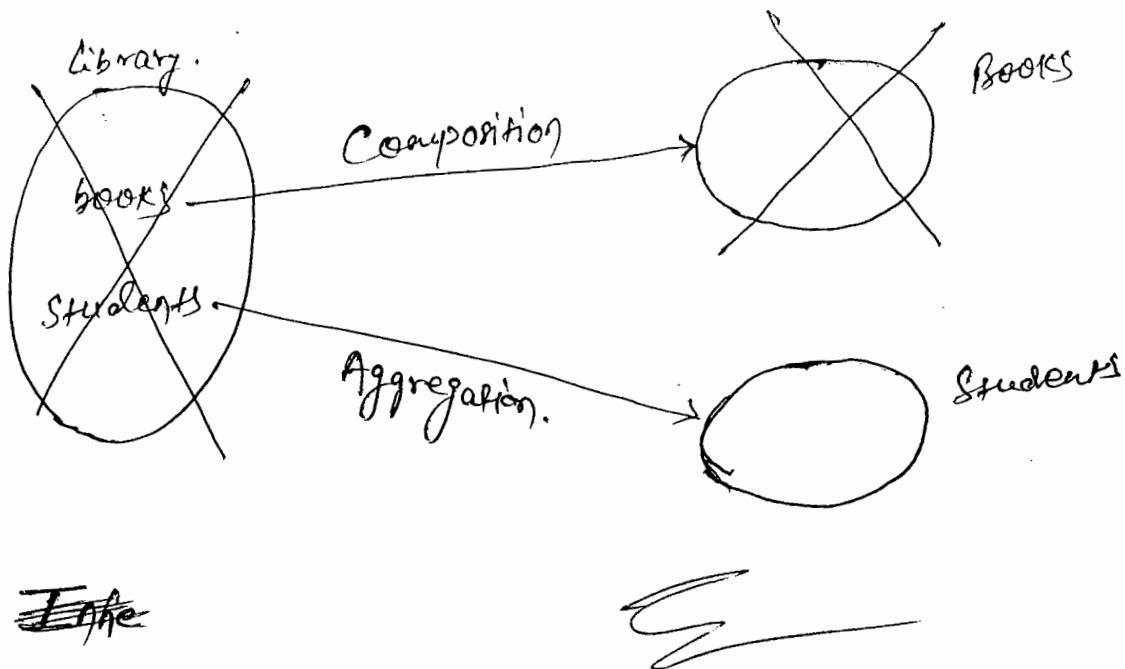
Example:-

If we consider the entities, like library, Books and Students then library contains books and Story. In this case if we close library, then there is no chance of existing books, but there is a chance of existing

Students may be e.g. class room or may be e.g. playground.

→ In the above example, strong association is existed between Library and Books, so that this association is called as "Composition".

→ In the above example, weak association is existed between Library and Students, so that this association is called as "Aggregation".



Inheritance in Java:-

If we want to prepare a software for an enterprise then we have to represent all the employee levels in the software, where the employee levels may be Manager, Accountant, Engineer.

To represent the employee levels we have to use classes.

Ex:-

```

Class Manager
{
    String eid;
    String ename;
    - - - -
    void getEmpInfo()
    - - - -
}

```



```

Class Accountant
{
    String eid;
    String ename;
    - - - -
    void getEmpInfo()
    - - - -
}

```



```

Class Engineer
{
    String eid;
    String ename;
    - - - -
    void getEmpInfo()
    - - - -
}

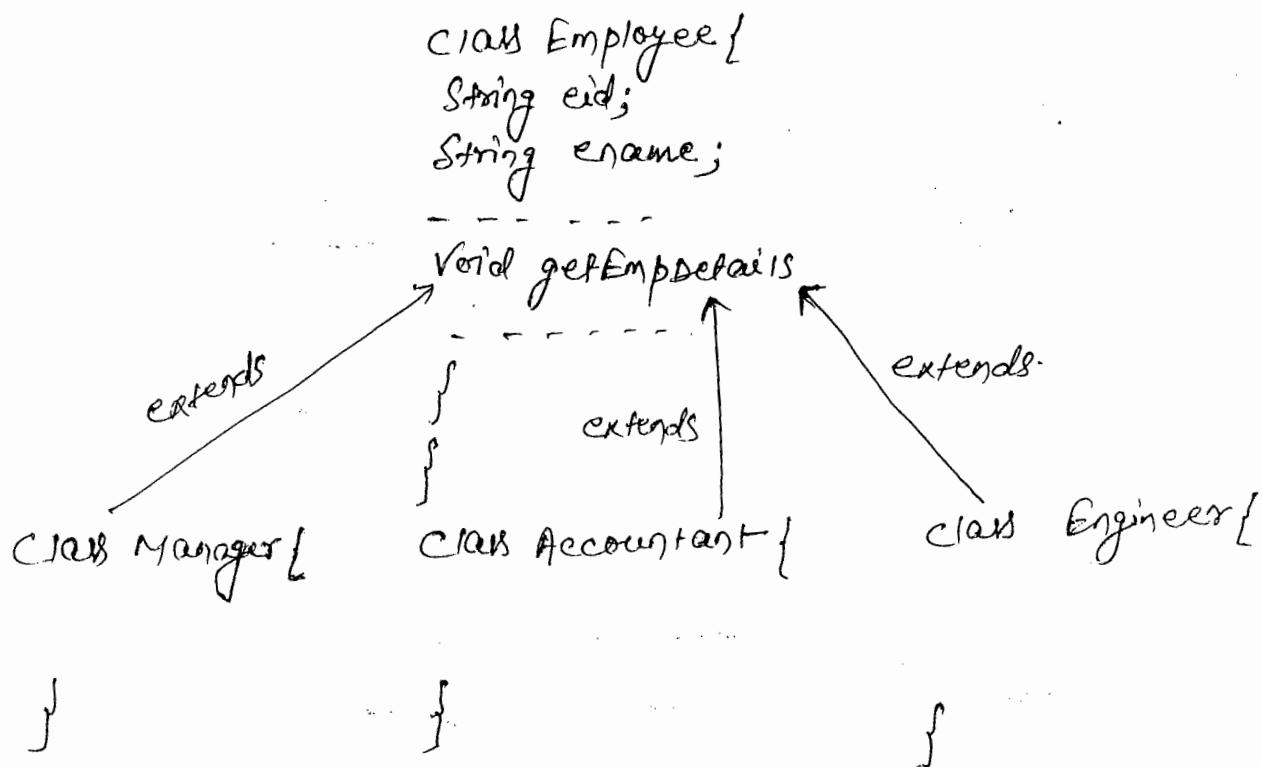
```

→ If we use the above approach in java apps then code duplication or code reusability, redundancy will be increased. It is not suggestible in application development.

In the above context, to reduce data redundancy, and to improve code reusability we have to use "Inheritance".

In case of Inheritance, we have to take a general class representing all the employee levels, where we have to declare all the common variables and methods then we have to extend this general class to all the classes where we want to reuse variables and methods.

→ The process of getting variables and Methods from one Class (Super class) to another Class (Sub class) is called as Inheritance.



The main objective of Inheritance is "Code Reusability".

→ In the case of Inheritance, if we declare variables and methods in the super class then we are able to access the members directly in the sub classes without creating object for the super class. If we declare variables and methods in the sub class then we are unable to access them directly in the super class.

If we declare reference variable for super class then by using that reference variable we are able to access only super class members. We are unable to access sub class own members. If we declare reference variable for sub class then by using that reference variable we are able to access both super class members and sub class members.

Ex:-

```

class A
{
    int i=10;
    void m1()
    {
        System.out.println("m1-A");
    }
    // System.out.println(j); → Error
    // m2(); → Error
}

```

Class B extends A → super class

```

class B extends A → super class
{
    int j=20;
    void m2()
    {
        System.out.println("m2-B");
        System.out.println(i);
        m1();
    }
}

```

Class Test

```

class Test
{
    public void main(String[] args)
    {
        A a = new A();
        System.out.println(a.i);
        a.m1();
        // System.out.println(a.j); → Error
        // a.m2(); → Error
    }
}

```

→ These are the sub class members and it is unable to access in super class. Because if we declare variables and methods in the sub class then we are unable to access them directly in the super class.

A/ Here, we are creating object and taking reference variable if we are only able to call super class A. Then by using this reference variable we can access super class B & sub class. Here for super class B & sub class, to access reference variable will occur. Here, we are creating object a and class A. Then it will occur. Here, we are creating object b and class B. Because a.i and b.i are accessing object a. They are also accessible because they are both super class objects. Sub class inherits both super class members and sub class members.

```

B b = new B();
System.out.println(b.i);
System.out.println(b.j);
b.m1();
b.m2();
}
}

```

→ Here, we are creating object for sub class B & sub class. Both class members are accessible for object b. Because they are both super class members and sub class members.

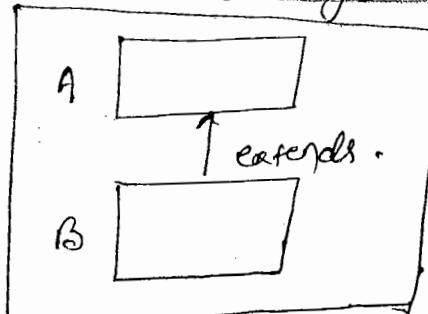
As per the basic level of Inheritance there are two types of Inheritance.

- ① Single Inheritance
- ② Multiple Inheritance.

① Single Inheritance:-

The process of getting variables and methods from only one super class to one or more no. of sub classes is called as Single Inheritance.

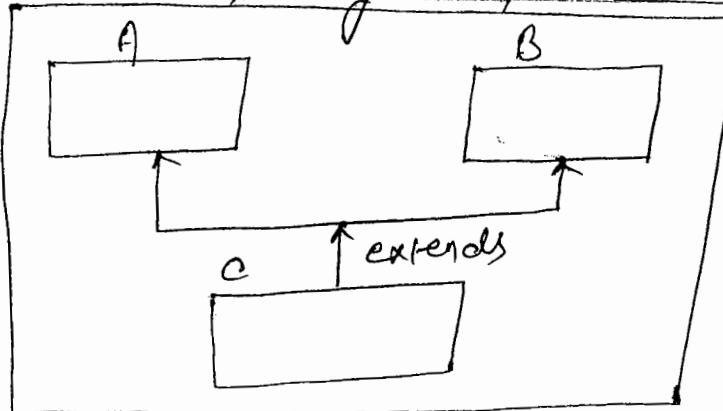
java is able to allow Single Inheritance.



② Multiple Inheritance:-

The process of getting variables and methods from more than one super classes to one or more no. of sub classes is called as Multiple Inheritance.

java is not supporting Multiple Inheritance.



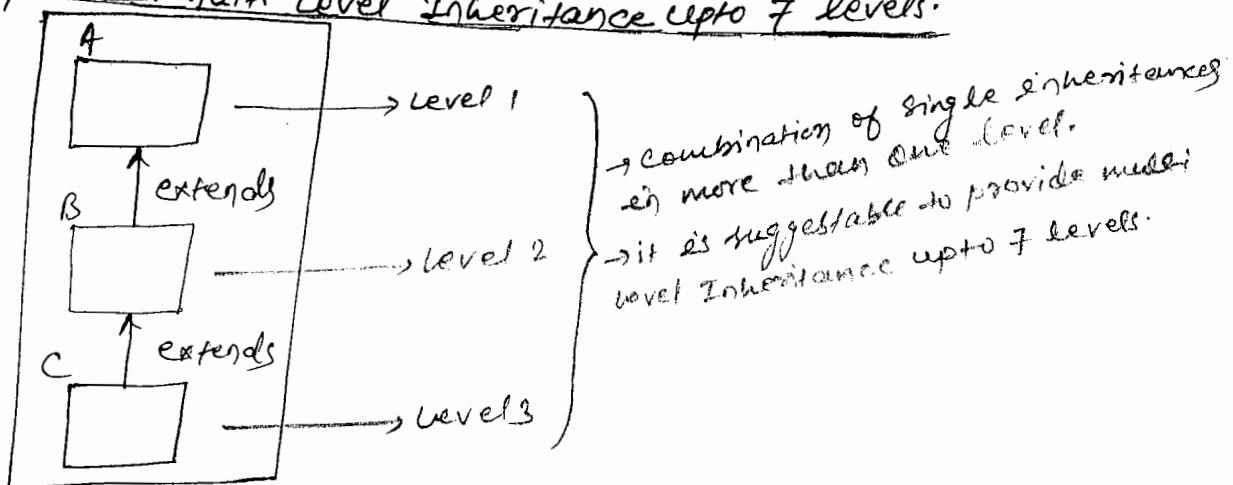
On the basis of Single and Multiple Inheritances, there are three more Inheritances.

- ① Multi level Inheritance
- ② Hierarchical Inheritance
- ③ Hybrid Inheritance.

① Multi Level Inheritance:-

It is the combination of single Inheritances in more than one level.

This inheritance is supported by java and it is suggestable to provide multi level Inheritance upto 7 levels.



② Hierarchical Inheritance:-

→ It is a combination of single Inheritance in a particular Structure like binary, tree, ...

This inheritance is supported by JAVA.



P.T.O

18-09-2015

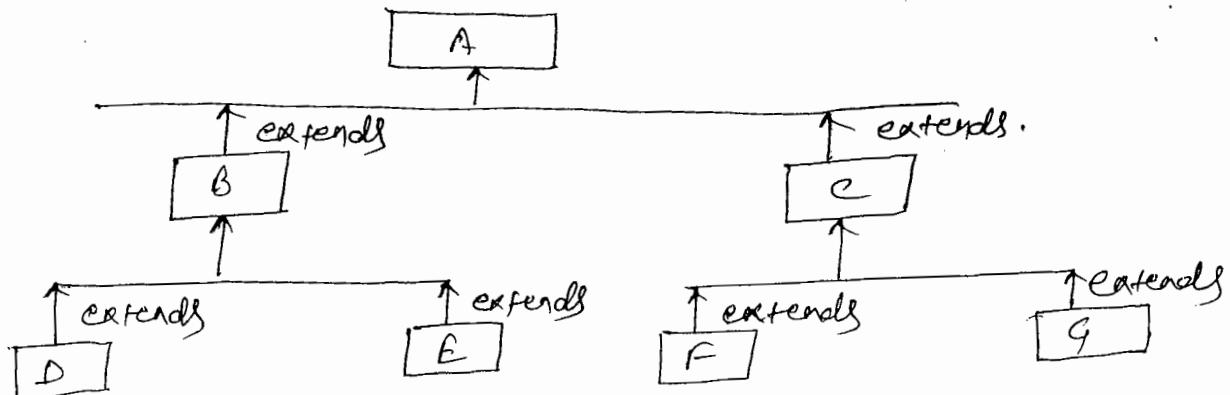
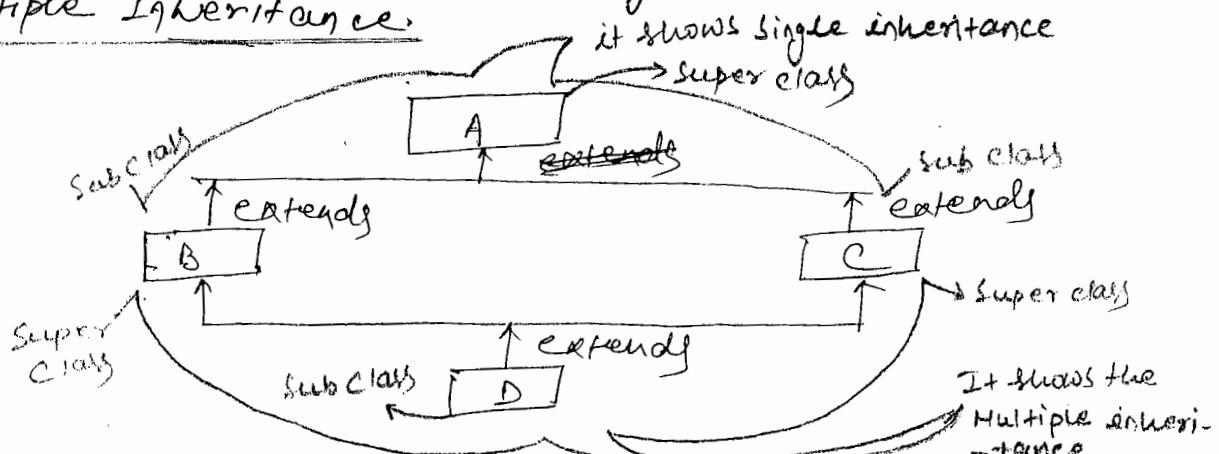


fig:- Hierarchical Inheritance.

Hybrid Inheritance:-

→ It is the combination of Single Inheritance and Multiple Inheritance.



→ This inheritance is not supported by Java.

Ex: (for showing single Inheritance by One Super class and two (more than one) super class.)
Class Employee → Super class.

```
{  
    String eid;  
    String name;  
    float sal;  
    String addr;
```

```
public void getEmpDetails() {  
    getEmpDetails declared  
}
```

```

S.O.Pln ("Employee Id :" + eid);
S.O.Pln ("Employee Name :" + ename);
S.O.Pln ("Employee Salary :" + esal);
S.O.Pln ("Employee Address :" + eaddr);
}
}

```

Class Manager extends Employee.

```

{
    Manager (String eid1, String ename1, float esal1, String eaddr1) → parameterized constructor
        of class Manager
}

```

```

    eid = eid1;
    ename = ename1;
    esal = esal1;
    eaddr = eaddr1;
}

```

```

public void getManagerDetails()
{
}

```

```

    S.O.Pln ("Manager Details");
    S.O.Pln ("-----");
}

```

getEmpDetails(); → It will call for getEmpDetails()

```

}
}

```

Class Accountant extends Employee.

```

{
    Accountant (String eid1, String ename1, float esal1, String eaddr1) → parameterized constructor of class
        Accountant
}

```

```

    eid = eid1;
    ename = ename1;
    esal = esal1;
    eaddr = eaddr1;
}

```

```

public void getAccountantDetails() getAccountantDetails()
{
    System.out.println("Account Details");
    System.out.println("-----");
    getEmpDetails(); → It will call for getEmpDetails()
}

```

from here it will print
the Account details of Employee.

Class Inheritance → Main Class.

```

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

```

Manager m = new Manager("E-111", "AAA", 50000, "Hyd");

m.getManagerDetails(); → Here, this method will access sub class
System.out.println(); (class Manager), so it will able to access super
class (Employee class) also.

Accountant acc = new Accountant("E-222", "BBB", 25000, "Hyd");

acc.getAccountantDetails(); → Here, this method access the sub
class (class Accountant), so it will able to access
super (Employee class) class also.

Static Context Inheritance

→ In java applications, static context is represented in the
form of the following elements.

- ① Static Variables
- ② Static Methods
- ③ Static Blocks

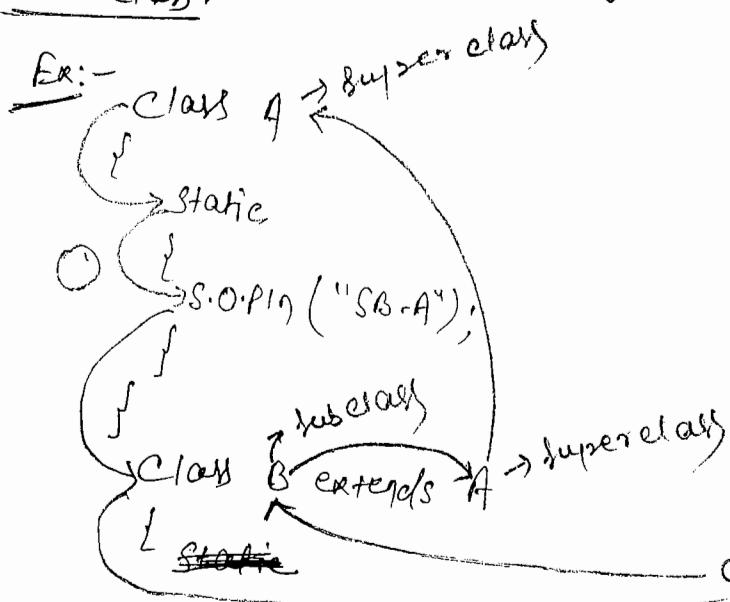
In java application, static context is recognized and executed at the time of loading the respective class bytecode to the memory.

In case of Inheritance, if we create Object for sub class then JVM has to execute sub class constructor, before this JVM has to load sub class bytecode to the memory.

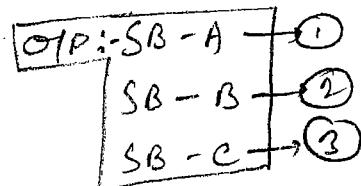
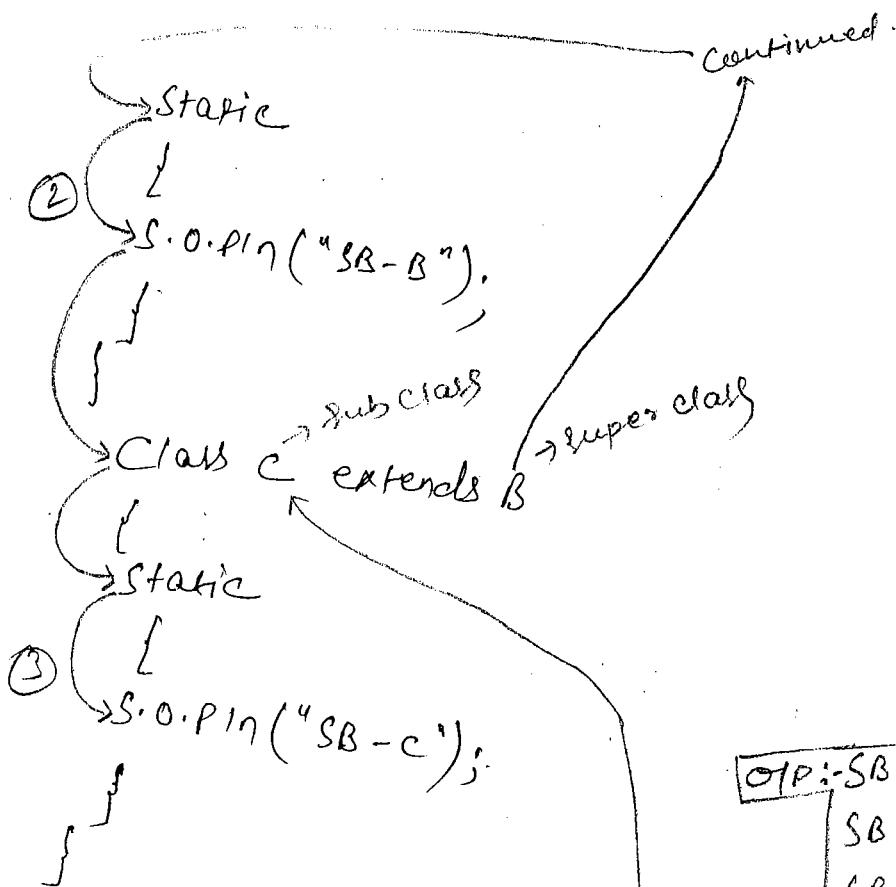
In case of Inheritance, when we are trying to load sub class bytecode to the memory, first JVM will load super class bytecode then only JVM will load sub class bytecode.

Therefore, in case of Inheritance JVM will load all the bytecode to the memory in super class to sub class order.

If we provide Static context, in all the classes which are available in Inheritance then JVM will execute Static Context automatically at the time of loading of respective class bytecode to the memory that is from super class to sub class.



continued .



Class Test → Main class.

{
 PSVM (String [] args)

{
 C = new C(); } → here, it will accept the class C

Ex

O/P:-

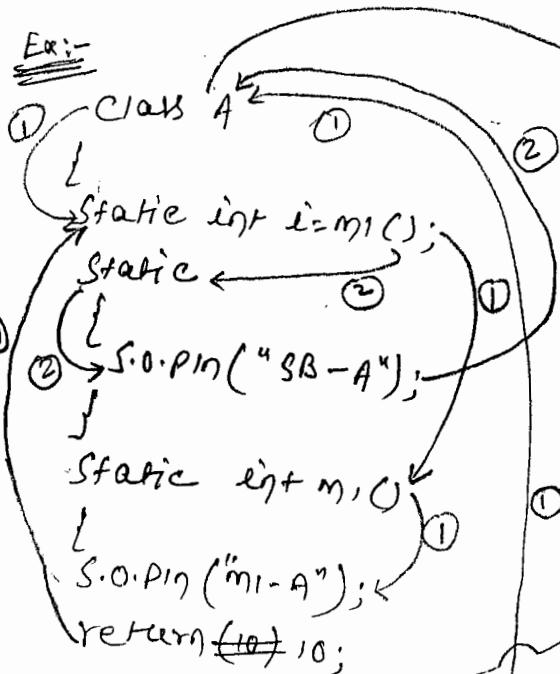
SB-A

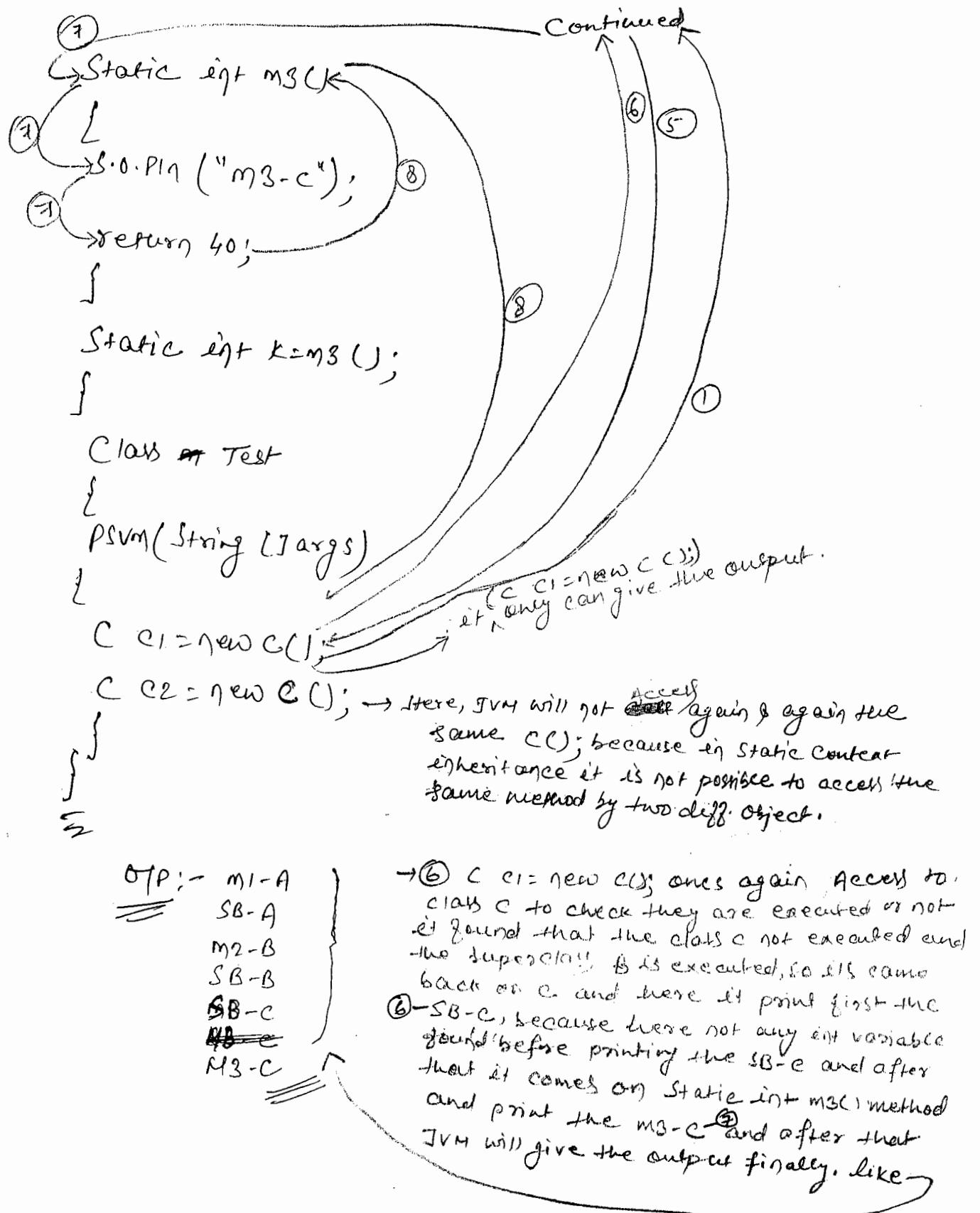
SB-B

SB-C.

==

(121)

Ex:-



Instance Context in Inheritance:-

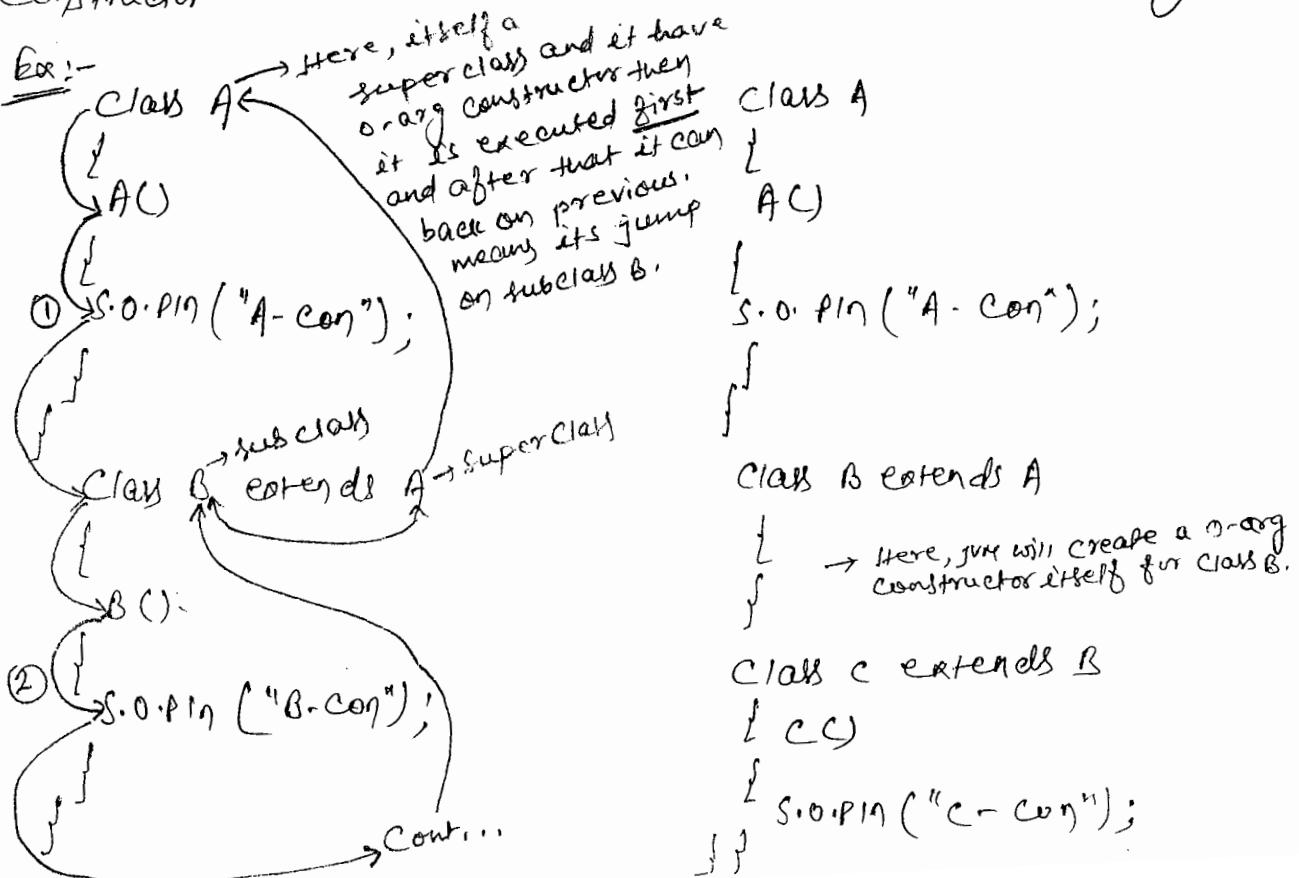
In java applications, instance context is represented in the form of the following three elements.

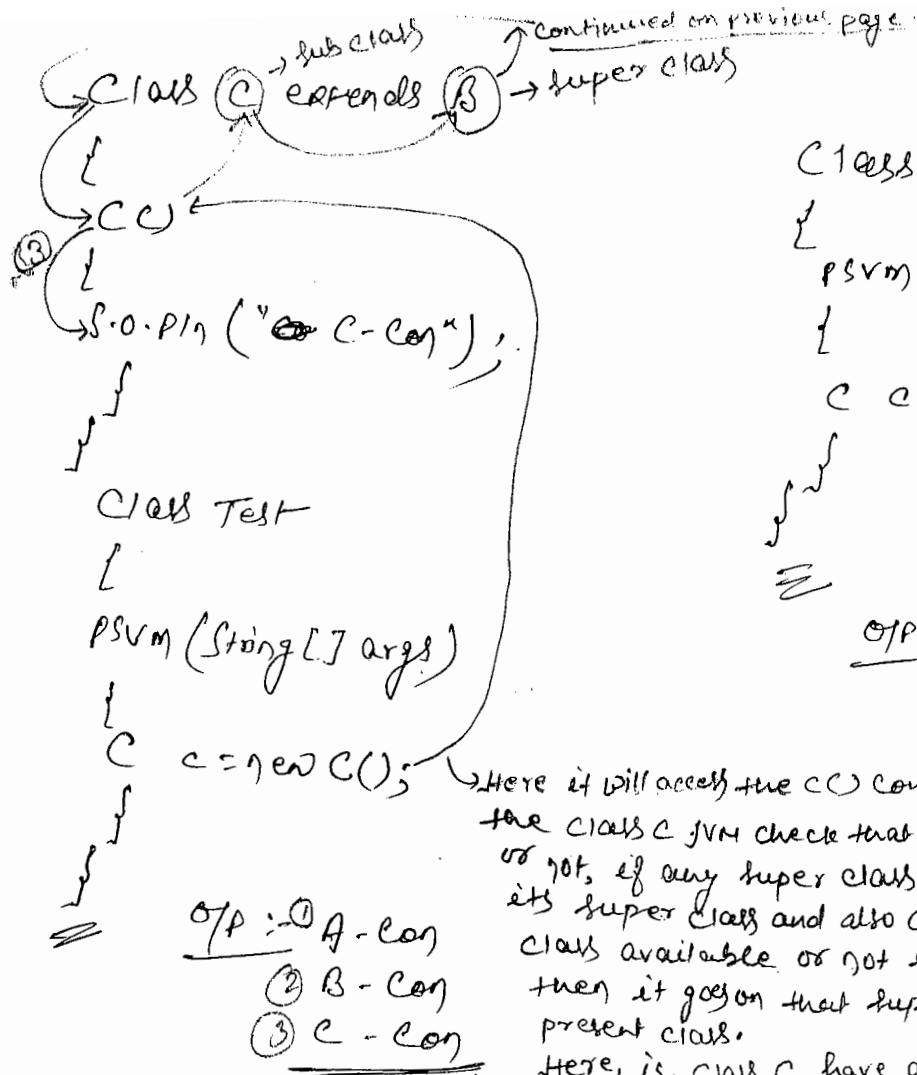
- (1) Instance Variables
- (2) Instance Methods
- (3) Instance Blocks.

In java applications, instance context will be recognized and executed just before executing the respective class constructor.

→ In case of Inheritance, if we create an object for the sub class then JVM has to access sub class constructor but before using sub class constructor JVM must access ^{Super class} 0-arg constructor.

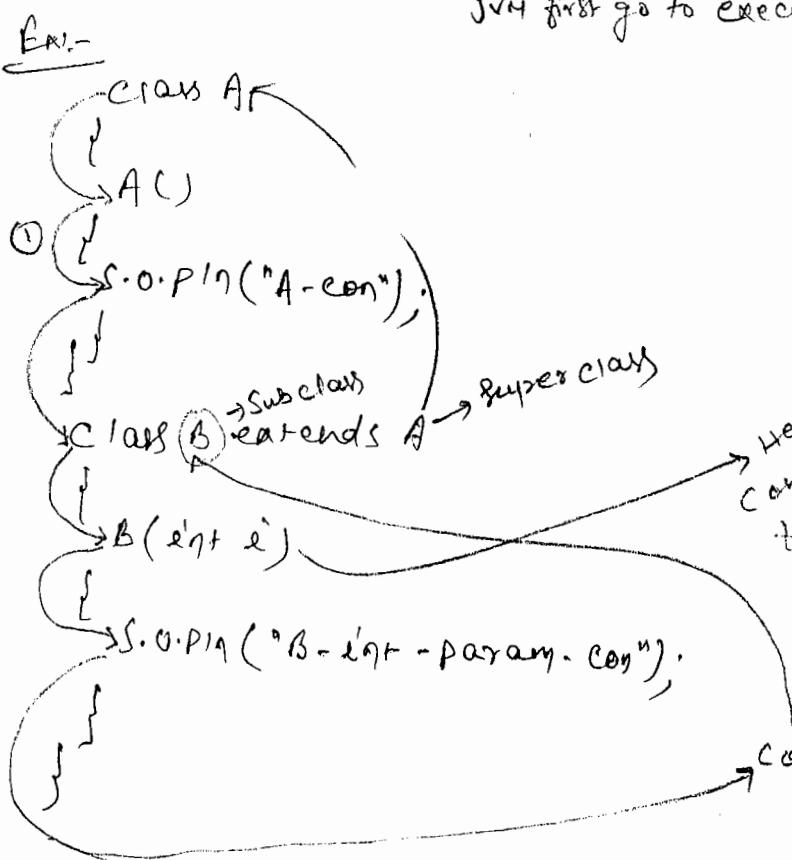
In case of Inheritance, Constructors execution order is from super class to sub class when we are accessing sub class constructor, but in super class JVM must access only 0-arg constructor.





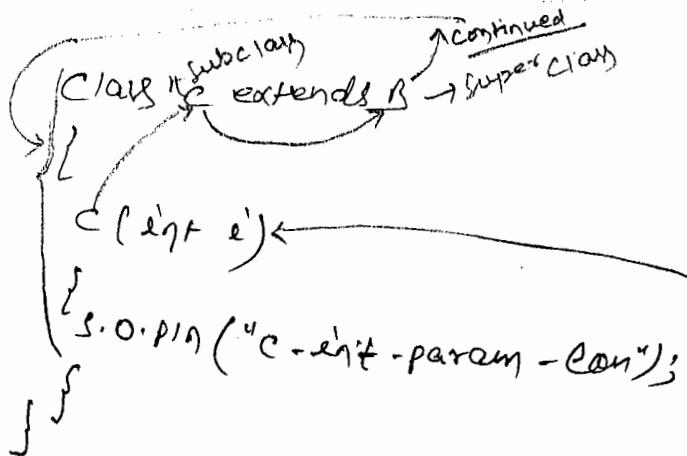
Here it will access the C() constructor but before execute the class C JVM check that there is any super class or not, if any super class available then it first go to its super class and also checked there is any super class available or not if any super class is there then it goes on that super class otherwise print that present class.

Here, is, Class C have a super class (Class B) then JVM first go to execute the Class B super class.



Here, it is parameterized constructor and it must have a arg for execute the class C, so it can't execute the class C with parameterized constructor so from here only the compilation error will be occurred.

Continued. . .



Class Test

PSVM (String [J args])

C C = new C();

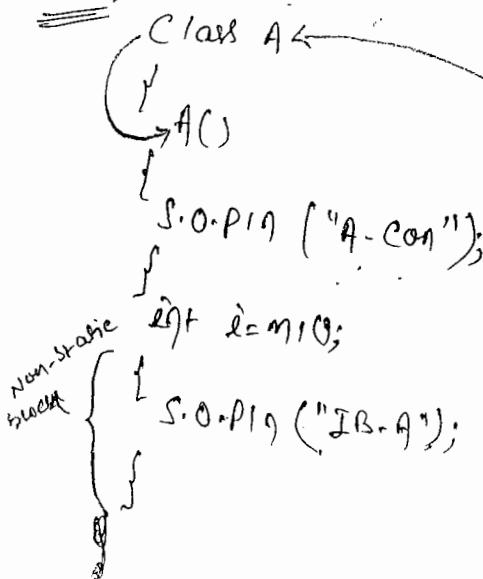
Status:-

CF (Here, is compilation error occurs to sub class, then super class in instance context, here, class A have a-arg constructor for execute sub class which is class B, But class B have not any a-arg constructor for execute his sub class (class C), so that there is compilation error must be occurred at the time of compilation.)

Reason:- In super class B 0-Arg Constructor is required, but it is not provided.

In case of Inheritance, if we provide instance Context at all the classes then JVM will execute the instance context just before executing the respective class constructor.

Ex:-



(123)

→ Here, the object access to class C (10) → C(int i) but the class C have super class (class B) to inherit purpose, and also check that class B have any super class or not, if not then from class A and end with class C (Means super class to sub class to sub class).

A class is the super class of when jvm will have not any other super class, then super class must have a-arg constructor for execute sub class which is class B, But class B have not any a-arg constructor for execute his sub class (class C), so that there is compilation error must be occurred at the time of compilation.)

doubt

Continued.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayagar Bakery
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Continued

```
if m1()
{
    Non-Static Method
    S.O.P("m1-A");
    return 10;
}

Class B {
    extends A { SuperClass }
}

if m2()
{
    S.O.P("m2-B");
    return 30;
}

S.O.P("IB-B");

B()
{
    S.O.P("B-con");
}

if j = m2();
{
    Class C {
        extends B { SuperClass }
    }
}

S.O.P("IB-C");

if m3()
{}
```

continued

continued

```

S. O. P10 ("m3 - c");
return 40;
}
C C) {
{
S. O. P10 ("c - con");
}
right IC = M3();
}

```

class Test

```

{
psvn (String [] args)
{

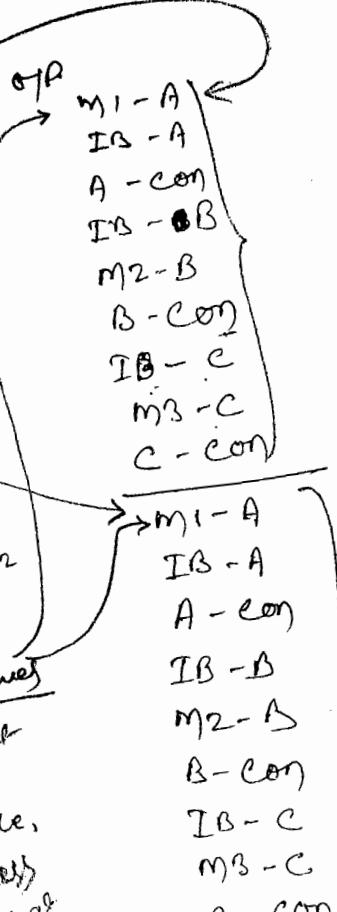
```

```

IC C1 = new C();
IC C2 = new C();
}
}

```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.



In Instance Context, two objects c_1, c_2 can access the same constructor one by one and the JVM will give the output by two times for these two objects but the output will be same.

But in Static Context it cannot possible, because in static context JVM can't detect the same constructor again and so, that if this situation occurred then JVM will help to print the output only one time.

Ex:

Class A

{

{

S.O.P19 ("IB-A"); → instance context

static int m1();

{

S.O.P19 ("m1-A");

return 10;

}

int i = m2();

Static

{

S.O.P19 ("SB-A"); → static context

}

int j = m2();

{

S.O.P19 ("m2-A");

return 20;

}

Static int j = m1(); → static variable

AC

{

S.O.P19 ("A-con");

{

Class B extends A

{

Static

{

S.O.P19 ("SB-B");

{

```

    {
        S.O.P1n ("IB-B");
    }

    static ejt m3()
    {
        S.O.P1n ("m3-B");
        return 30;
    }

    ejt m4()
    {
        S.O.P1n ("m4-B");
        return 40;
    }

    ejt k=m5();
    static ejt l=m3();
    B()
    {
        S.O.P1n ("B-C07");
    }
}

Class C extends B
{
    ejt m=m5();
    ejt m5()
    {
        S.O.P1n ("m5-C");
        return 50;
    }

    S.O.P1n ("IB-C");
}

C()

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

```

S.O.P("C.con");
{
Static
{
S.O.P("SB-C");
{
Static int n=m6();
Static int m6()
{
S.O.P("m6-C");
return 60;
}
}
}

```

```

Class Test
{
PSVM(String[] args)
{
C c1=new C();
C
C c2=new C();
}
}

```

O/P:-

SB-A
M1-A
SB-B
M3-B
SB-C
M6-C
IB-A
M2-A
A-con
IB-B
M4-B
B-con
M5-C
IB-C
C-con

IB-A
M2-A
A-con
IB-B
M4-B
B-con
M5-C
IB-C
C-con

Super keyword:-

'Super' is a java keyword, it can be used to represent Super Class Object from Sub class.

In java applications, we are able to utilize 'Super' keyword in the following three ways.

- ① To refer Super Class Variables
- ② To refer Super Class Methods
- ③ To refer Super Class Constructors

① To refer Super Class Variables:-

If we want to refer larger class variables by using 'Super' keyword then we have to use the following syntax:

`Super.Var_Name;`

Note:- In java applications, when we have same set of variables at local, at current class and at super class, where to refer Super class variables over local variables and current class variables then we have to use 'super' keyword.

Ex:-

Class A
If there static available then
also same output will come.

{ int i=10; } → class level or Globalized or
int j=20; } → super class variable or parent Class

}

→ sub class

Class B extends A → super class

{ int i=30; } → local variable or child class
int j=40; } → here it is class level variable, because of parameterized
constructor and this statement will print it only.
because it is the present class after executing
the local variable
B(int i, int j) → parameterized constructor

S.o.pn("Local vars : " + i + " " + j);

(50,60) here, the local variable is local variable which is accepted by object and here class B constructor have parameterized so, the value of constructor is local variable here.

S.o.pn("Class Lvl vars : " + this.i + " " + this.j);

(30,40) - this statement will store the value in current class only. And here the current class is class B.

S.o.pn("Super Class Var : " + super.i + " " + super.j);

→ It will print the super class (class A) value because class A is the super class of sub class (class B). And it haven't any super class. And the "super" keyword directly print the super class value without checking by other sub classes.

}

Class Test

{

PSVM (String[] args)

}

B b = new B(50,60);

O/P :-	50	60
30	40	
10	20	

When the object will access the constructor there is B constructor have some parameters of int type and O/P :- the object have its value. The object will store this value into parameterized constructor B(int i, int j) and it is printed as local variable of sub class B.

(2) To refer Super Class methods :-

If we want to refer super class method by using 'super' keyword then we have to use the following syntax.

Super.method_name(Param - List);

Note:- In java applications, when we have same method at super class and at current class then to access super class method over current class method we have to use 'super' keyword.

Ex:-

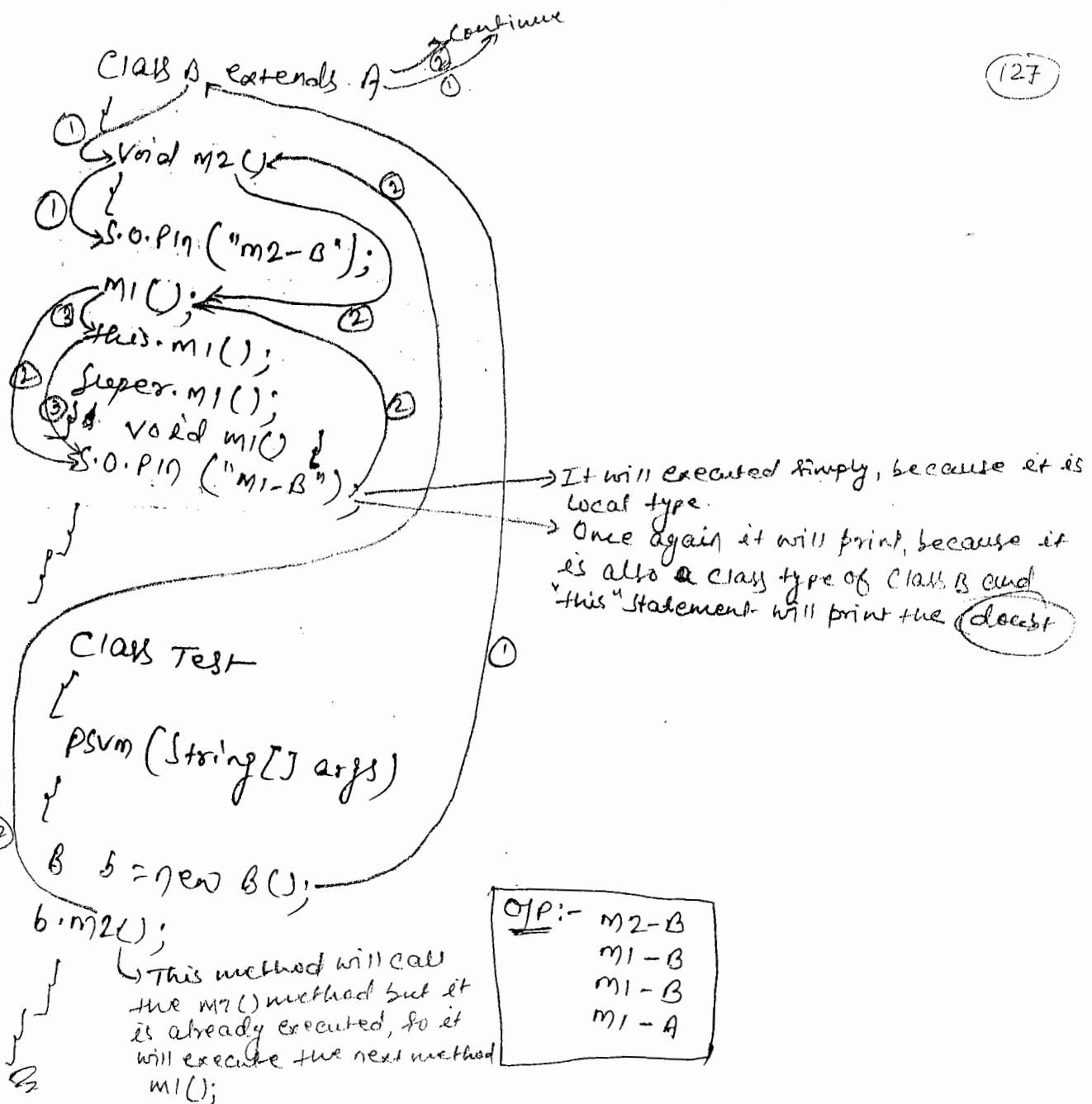
Class A

{

Void m1()

{ S.o.pn("m1 - A"); }

}



③ To refer Super Class Constructors:-

If we want to refer super class Constructors by using `Super` keyword from sub class then we have to use the following syntax.

`Super([param_list]);`

Note:- In general, In inheritance JVM will execute 0-arg Constructor in Super class, before executing Sub class Constructor.

In this context, if we want to execute parameterized constructor in the super class before sub class constructor then we have to use 'super' keyword.]

→ If we want to use super keyword to refer super class constructor then the respective super statement must be provided as first statement inside the sub class constructors, not in normal java methods. If we violate this rule then compiler will give an error like "Call to super must be first statement in the Constructors".

~~Ex~~

Class A

{
A()
}

{
S.O.Pln("A-con");
}

A (int i)

{
S.O.Pln("A-int-param-con");
}

Class B extends A

{
B()
}

Super();

{
S.O.Pln("B-con");
}

o/p: A-int-param-con
B-con

Class Test

{
PSVM (String [] args)
}

B b=new B();
}

Q

Ex:-

Class A

```

    {
        A (int i) → Error (if super (10); method is not define as a first statement
                           of present constructor, then it also can't avoid
                           that A (int i) parameterized constructor)
                           → when super (10) is a first statement of constructor
                           then A (int i) is correct
    }
    {
        S.O.Pln ("A - int - param - Con");
    }
}

```

Class B extends A

{

B()

{

S.O.Pln ("B - Con").

Error - Scopes (10); → It must be
must statement
in constructor

Class Test

{

PSVM (String [J args)

{

B b = new B();

{

3

Status:- CE.

Ex:-

Class A

{

A (int i)

{

S.O.Pln ("A - int - param - con");

{

Class B extends A

{

Virtual m1 ()

{

super (10); → we can't write
inside the
method

S.O.Pln ("m1 - B");

{

Class Test

{

PSVM (String [J args)

{

B b = new B();

b.m1 ();

{}

→ In java applications, we are able to refer only one super class constructor by using 'super' state keyword from single sub class constructor, if it's not possible to access more than one super class constructor from a sub class constructor, because Super Statement must be first statement.

Ex:-

Class A {

A() {

System.out.println("A-con");
}

A(int i) {

System.out.println("A-int-param-con");
}

Class B extends A

B() {

System.out.println("B");

It can't be possible to access more than one super class constructor from a sub class constructor.

because Super class must be first statement.

Super(); → because Super class and here for Super(); it will be executed of the constructor and here for Super(); it can't be executed.

Super(10); → executed but for Super(10); it can't be executed because it is not the first statement. So, here compilation error will be occurred.

System.out.println("B-con");
}

Class Test {

public static void main(String[] args) {

B b = new B();

}

States:- Compilation error.

In case of Inheritance, if we compile java program then compiler will perform the following two actions w.r.t the constructors in all the classes.

- ① Compiler will go to each and every class and checks whether any constructor is existed or not explicitly, if no constructor is provided (existed) then compiler will provide default constructor. If any user defined constructor is existed then compiler will not provide default constructor.
- ② Compiler will go to each and every class and checks ^{Constructor} ~~and checks~~ in each and every class and checks any "super(--)" is existed or not explicitly in order to access super class constructor, if no "super(--)" statement is existed explicitly then compiler will add "super()" statement as the first statement in the constructor in order to execute super-class no-arg constructor. If any "super(--)" statement is existed explicitly then compiler will not add "super(--)" statement.

Ex:- (internal concept)

```

Class A{
    A(){}
    S.O.P("A-con");
}
}

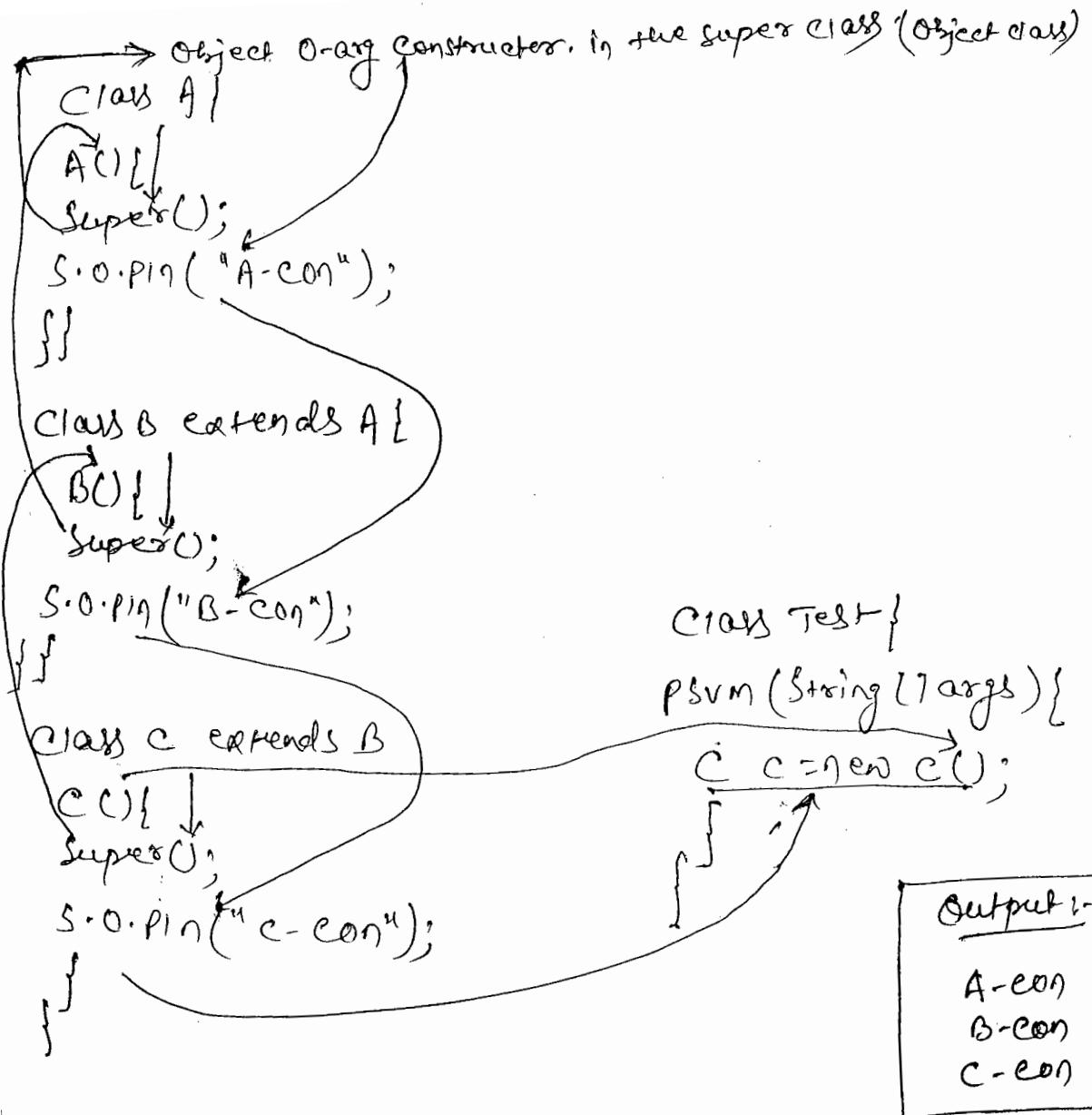
Class B extends A{
    B(){}
    S.O.P("B-con");
}

Class C extends B{
    C(){}
    S.O.P("C-con");
}

Class Test{
    Psvm(String[] args){
        C c = new C();
    }
}

```

Compilation → Cont...!



Ex:-

```

class A{
    A(){
        S.O.PIn("A-con");
    }
}

class B extends A{
    B(int i){
        S.O.PIn("B-con");
    }
}

class C extends B{
    C(){
        S.O.PIn("C-con");
    }
}

```

```

class Test{
    psvm(String[] args){
        C c = new C();
    }
}

```

Compilation → Cont...

=

Compiling Continued.

```
Class Test {  
    Test(){  
        super();  
    }  
    public static void main(String[] args){  
        C c = new C();  
    }  
}
```

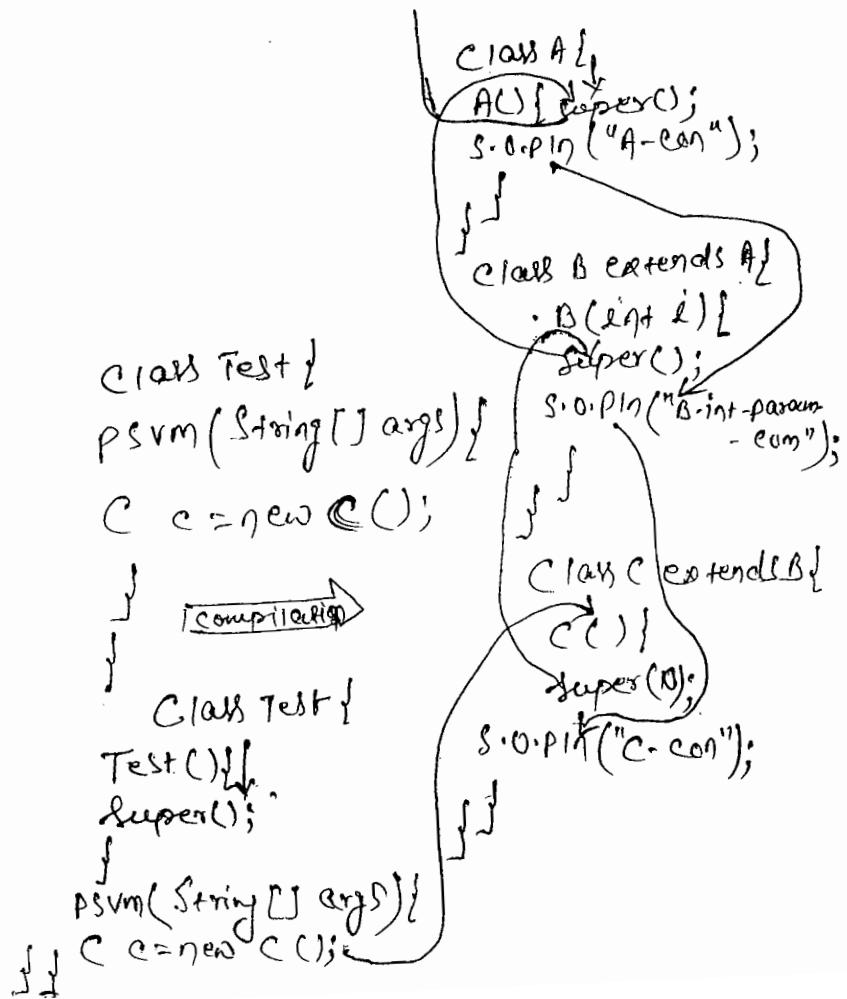
Status:- Compilation error

Class A {
 A(){
 super();
 System.out.println("A-con");
 }
}
Class B extends A {
 B(int i){ →
 Super();
 System.out.println("B-con");
 }
}
Class C extends B {
 C(){
 Super();
 System.out.println("C-con");
 }
}

Reason:- 0-arg constructor is required in Class B.

Ex:-

```
Class A{  
    A(){  
        super();  
        System.out.println("A-con");  
    }  
}  
Class B extends Class A {  
    B(int i){  
        System.out.println("B-int-param-con");  
    }  
}  
Class C extends B {  
    C(){  
        Super(10);  
        System.out.println("C-con");  
    }  
}
```



Class level type Casting:-

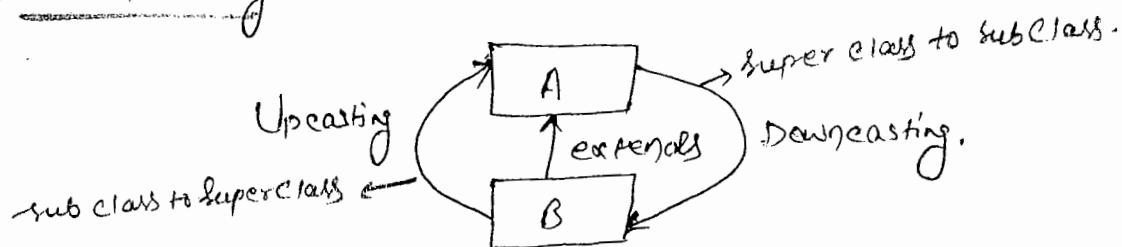
The process of converting data from one user defined data type to another user defined data type is called as Class level type casting or User defined data types Type casting.

To perform user defined data types type casting we must provide either "extends" relationship or "implements" relationship between two user defined data types.

There are two types of User defined data types type casting.

① Upcasting

② Down Casting.



① Upcasting:-

The process of converting data from sub Class type to Super Class type is called as Upcasting.

In java applications, to perform upcasting we have to assign sub class reference variable to super class reference variable without having any cast operator explicitly.

Ex:-

```
Class A { }  
Class B extends A {  
    Class Test {  
        public void main(String[] args) {
```

```
B b = new B();  
A a = b;
```

24/11/2015
131

Ex:-

```
class A {  
    void m1() {  
        S.O.P("m1-A");  
    }  
}
```

Class B extends A

```
{  
    void m2() {  
        S.O.P("m2-B");  
    }  
}
```

Class Test {

```
PSVM(String [] args)  
{
```

```
B b = new B();
```

```
b.m1();
```

```
b.m2();
```

```
A a = b; // → ①
```

```
a.m1();
```

```
}
```

=

When we compile the above code [label-1] then compiler will check whether right side variable [b] data type is compatible with left side variable [a] data type or not.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

If right side variable data type is not compatible with left side variable data type then compiler will rise an error like "incompatible types error". If right side variable data type is compatible with left side variable data type then compiler will not rise any error and compiler will not perform any type casting.

When we execute the above code, [label-1] then JVM will perform the following two actions.

- ① JVM will convert right side variable data type to left side variable data type implicitly (upcasting).
- ② JVM will copy right side variable value to left side variable.

Note:- In java applications, always, sub class types are compatible with super class types, so that we can assign directly sub class reference variable to Super class reference variable.

In java applications super class types are not compatible with sub class types directly, so, that we are unable to assign super class reference variable to sub class reference variable directly.

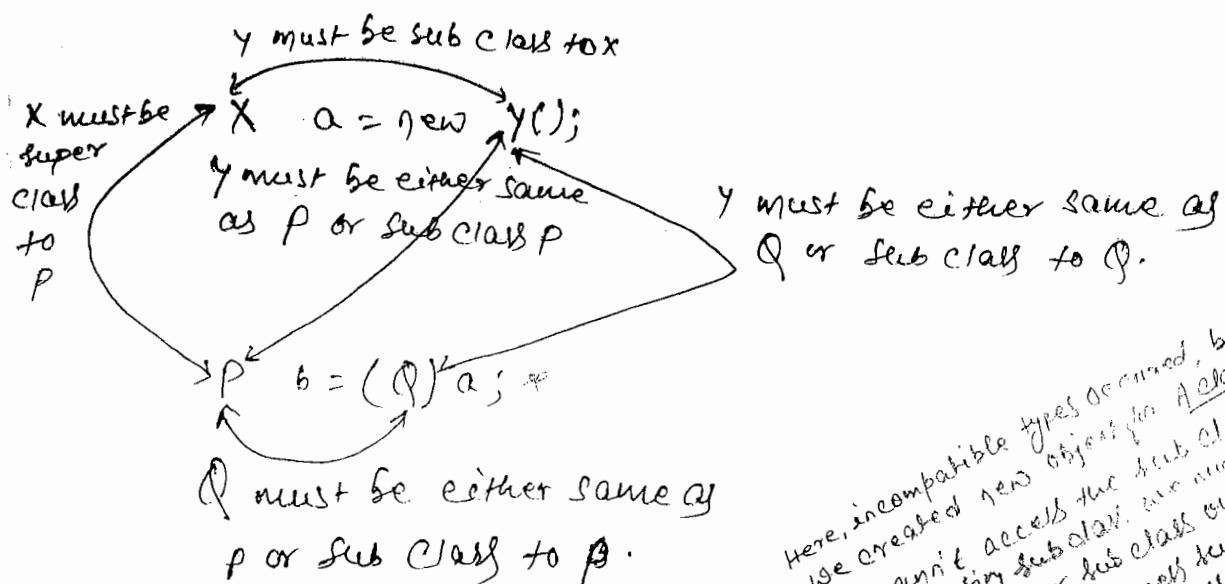
If we want to assign super class reference variable to sub class reference variable then we have to use cast operator explicitly that is Downcasting.

2

Downcasting:-

The process of converting data from super class type to sub class type is called as downcasting.

To perform downcasting we have to use the following structure.



Here, incompatible types occurred, because we created new object for A class and it cannot access the sub class (B class) for accessing sub class, we must create new object for sub class only and sub class can access super class (A class) but super class (A class) can't access the sub class.

Ex:-

```

class A {
    void m1() {
        System.out.println("m1-A");
    }
}
  
```

Class B extends A

```
{
    void m2()
}
```

```
{ System.out.println ("m2-B"); }
```

→ Here also we can't access sub class by creating object for super class.

→ Here, we created the new object for sub class and it can access the super class.

→ Here, we are using downcasting by converting the super class ref.

→ Here, I can access both super and sub class.

PSVM (String [] args) {

A a = new A();

B b = a; → Incompatible types

A a = new A();

B b = (B)a; → ClassCastException

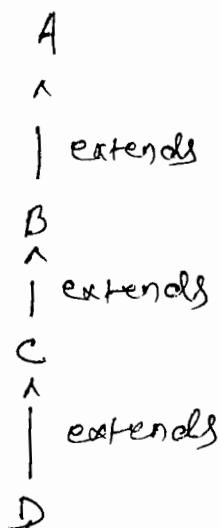
- A a = new B();

a.m1();

B b = (B)a; // → Downcasting

b.m1(); b.m2(); } }

- ① In java applications, Only sub class types are compatible with super class types, so that we can assign sub class reference variable to super class reference variable directly.
- ② In java apps, super class types are not compatible with sub class types directly, so, that we are unable to assign super class reference variables directly to sub class reference variables. If we assign super class reference variable directly to sub class reference variable then compiler will rise an error like "Incompatible type error".
- ③ In java apps we are able to keep sub class object reference value in super class reference variable, but, we are unable to keep super class Object reference value in sub class reference variable.
- ④ If we are trying to keep super class Object reference value in sub class reference variable then Java will rise an exception like "java.lang.ClassCastException".



Ex:- ①

A a=new A();
B b=q;

Status:- Compilation Error, Incompatible types.

→ Here, compilation error occur because of
super class object is created new object for super
class only and super class can't access sub class.
So, here, super class must have to create new object
for sub class to access both super and sub class.

②

A a=new A();
B b=(B)a;

Status:- No Compilation error, ClassCastException

③

A a=new B();
B b=(B)a;

Status:- No Compilation error, and No Exception.

④

A a=new C();
B b=(C)a;

Status:- No Compilation error and No Exception.

⑤

A a=new D();
B b=(C)a;

Status:- No Compilation error and No Exception

⑥

A a=new B();
B b=(C)a;

Status:- No Compilation error, ClassCastException,

Error:-
⑦

A $a = \text{new } C();$

B $b = \text{new } D(a);$

Status:- No compilation error, Classcast Exception.

⑧

B $b = \text{new } A();$

C $c = (D)b;$

Status:- Incompatible Types Error.

⑨

A $a = \text{new } D();$

D $d = (D)(C)(B)a;$

Status:- No compilation error, No exception.

⑩

A $a = \text{new } C();$

C $c = (D)(C)(B)a;$

Status:- No Compilation Error, Classcast Exception.

Note:- In java apps when we have requirement to access only super class members from the sub class object then we have to use casting. If we create sub class object then in the sub class object both super class members and sub class members are available, where to access only super class members not sub class members then we have to declare

25-09-2015

(134)

reference variable for super class type.

Ex:- A a=new B();

Uses-A Relationship:-

It is a relation between entities, where one entities will be used upto a particular action or behaviour of another entity.

To achieve USES-A relationship between entities we have to pass the respective entity class reference as parameters to another entity method.

Ex:-

Class Account

{

String accNo;

String accName;

String accType;

int balance;

Account(String accNo, String accName, String accType, int balance)

{

this.accNo = accNo;

this.accName = accName;

this.accType = accType;

this.balance = balance;

}

Class Transaction

{

```
public void withdraw(Account acc, int wdAmt)
```

```
}
```

```
acc.balance = acc.balance - wdAmt;
```

```
s.o.println("Transaction Details");
```

```
s.o.println("-----");
```

```
s.o.println("Account Number :" + acc.accNo);
```

```
s.o.println("Account Name :" + acc.accName);
```

```
s.o.println("Account Type :" + acc.accType);
```

```
s.o.println("Withdraw Amount :" + wdAmt);
```

```
s.o.println("Total Balance :" + acc.Amt);
```

```
}
```

```
Class Test
```

```
{
```

```
psvm(String [] args)
```

```
{
```

```
Account acc = new Account("abc123", "Durga", "Savings", 2500);
```

```
Transaction tx = new Transaction();
```

```
tx.withdraw(acc, 10000);
```

```
}
```

Polymorphism:-

Polymorphism is a greek word, where poly means many, and morphism means structures or forms.

If one thing is existed in more than one form then it is called as polymorphism.

→ The main objectives of polymorphism is flexibility in appn development.

There are two types of polymorphisms.

- ① Static Polymorphism
- ② Dynamic Polymorphism

① Static Polymorphism :-

If the polymorphism is existed at compilation time then that polymorphism is called Static polymorphism.

Ex:- Method Overloading.

② Dynamic Polymorphism:-

If the polymorphism is existed at runtime then that polymorphism is called as Dynamic Polymorphism.

Ex:- Method Overriding.

Method Overloading:-

The process of extending existed method functionality upto some new functionality is called as Method Overloading.

Overloading.

- If we declare more than one method with the same name and with different parameter list then it is called as method "Overloading".
- To perform Method Overloading we must provide diff. method signatures between all the methods, where diff. parameter list is available in either of the following ways.

- ① Difference in the no. of parameters.
- ② Difference in the parameter data types.
- ③ Difference in the parameter data type Order.

Ex:-

Class A

{

Void add(int i, int j)

{

S.O.P19(i+j);

}

Void add (float f1, float f2)

{

S.O.P19(f1+f2);

}

Void add (String Str1, String Str2)

{

S.O.P19(Str1+Str2);

}

here, there all are the method with the same name (i.e. void add) but each one have different parameters (i.e. int, float, String) so, here Method overloading occurred.

Class Test

```

    {
        PSVM (String [] args)
    }

    A a = new A();
    a.add(10, 20);
    a.add(22.22f, 33.33f);
    a.add("abc", "def");
}

```

Ex:-

```

class Employee
{
    public void gen_salary (int basic, float hra, int ta, float pf)
    {
        float sal = basic + ((basic * hra) / 100) + ta - ((basic * pf) / 100);
        System.out.println ("Salary :" + sal);
    }

    public void gen_salary (int basic, float hra, int ta, float pf,
                           int bonus)
    {
        float sal = basic + ((basic * hra) / 100) + ta - ((basic * pf) / 100) +
                    bonus;
        System.out.println ("Salary :" + sal);
    }
}

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

Class Test

```
{  
public static void main(String[] args)  
{  
    Employee emp = new Employee();  
    emp.genSalary(2000, 25.0f, 10000, 12.5f);  
    emp.genSalary(2000, 25.0f, 10000, 12.5f, 5000);  
}  
}
```

Method Overriding:-

The process of providing replacement for an existed method functionality with new functionality is called as method overriding.

In java applications, to perform method overriding we must require inheritance and we must provide same method prototype for both super class method and sub class method.

→ In method Overriding process, super class method functionality is existed method functionality and sub class method functionality is new functionality.

Continued:-

Ex:-

```

Class Loan → super class
{
    public float getIR()
    {
        return 5.0f;
    }
}

```

```

Class Gold_Loan → sub class
{
    extends Loan → super class
}

```

```

    public float getIR()
    {
        return 6.0f;
    }
}

```

OP:-
Gold Loan : 6.0
Study Loan : 12.5
Craft Loan : 3.0

```

Class Study_Loan → sub class
{
    extends Loan → super class
}

```

```

    public float getIR()
    {
        return 12.5f;
    }
}

```

Class Test

```

    PSVM (String [] args)
    {

```

```

        Loan gl = new Gold_Loan();
        System.out.println("Gold Loan IR :" + gl.getIR());
        Loan sl = new Study_Loan();
        System.out.println("Study Loan IR :" + sl.getIR());
        Loan cl = new Craft_Loan();
        System.out.println("Craft Loan IR :" + cl.getIR());
    }
}

```

```

Class Craft_Loan → sub class
{
    extends Loan → super class
}

```

```

    public float getIR()
    {
        return 3.0f;
    }
}

```

26-09-2015

Procedure to perform Method Overriding:-

- ① Declare a super class.
 - ② Declare a method in super class which we want to override.
 - ③ Declare sub class to the super class.
 - ④ Provide the same super class method with different implementation part in sub class.
 - ⑤ In main class, in main() method, we must create object for sub class and we have to declare reference variable for super class.
 - ⑥ Access super class method then we have to get output from sub class method.
- (Note:- In the above steps, to prove method overriding, we must access super class method but try must execute sub class method, to achieve this we must declare reference variable for super class but Object must be created for sub class.)

Ex:-

Class A

```
void m1()
```

```
{  
    S.O.P("m1-A");  
}
```

Class B extends A

```
{  
    void m1()
```

```
{  
    S.O.P("m1-B");  
}
```

Class Test

```
{  
    public void m1(String[] args)  
}
```

```
/*  
A a = new A();
```

```
a.m1();
```

Status: Method Overriding require subclass object.

```
/*  
B b = new B();
```

```
b.m1();
```

Status:- Method Overriding is performed when we create obj
of sub class but to prove method Overriding we must access
super class method, so that super class reference variable is
required.

```
*/
```

```
A a = new B();
```

```
a.m1();
```

```
}
```

Rules to perform method Overriding:-

① To perform method Overriding, we are unable to declare super class method as private.

Ex:-

```
Class A
```

```
private void m1() {  
    System.out.println("m1-A");  
}
```

```
Class B extends A {  
    void m1() {  
        S.O.Pln("m1-B");  
    }  
}
```

```
Class Test {  
    PSVM (String [] args) {  
        A a = new B();  
        a.m1();  
    }  
}
```

Status:- Compilation error. (m1() has private access in A)

a.m1()

"error"

~~Rule :-~~

If we want to Override super class method with sub class method then sub class method return type must be same as super class method return type.

Ex:-

```
Class A {  
    int m1() {  
        S.O.Pln("m1-A");  
        return 10;  
    }  
}
```

```
Class B extends A {  
    int m1() {  
        S.O.Pln("m1-B");  
        return 20;  
    }  
}
```

```
Class Test {  
    PSVM (String [] args) {  
        A a = new B();  
        a.m1();  
    }  
}
```

Status:- No compilation error
OP:- m1-B

~~Ex:-~~

```

class A{
    int m1(){
        S.O.P("m1-A");
        return 10;
    }
}

```

Class B extends A

```

}
byte m1(){ → Error.
    S.O.P("m1-B");
    return 30;
}

```

```

class Test{
    Psvm(String [] args){

```

```

        A a = new B();
    }
}

```

A. m1();

Z Status:- Compilation error.

→ m1() in B cannot override m1() in A
 ↗ ext m1() of
 ↗ return type int is not compatible with
 ↗ byte.

③ If we want to override super class method with sub class Method then super class method must not be declared as final irrespective of sub class method final.

~~Ex:-~~

```

class A{
    final void m1(){
        error. S.O.P("m1-A");
    }
}

```

Class B extends A

```
final void m1() {  
    System.out.println("m1-B");  
}
```

if we use the final keyword here; then there is no any compilation error occurred.

```
Class Test {  
    public static void main(String[] args) {  
        A a = new B();  
        a.m1();  
    }  
}
```

Status: Compilation Error [m1() in B cannot override m1() in A]
void m1()
^
Overridden method is final
} errors.

Ex:

```
Class A {  
    final void m1() {  
        System.out.println("m1-A");  
    }  
}
```

error for this "final" keyword only.
because it is not possible here.

```
Class B extends A {  
    final void m1() {  
        System.out.println("m1-B");  
    }  
}
```

error is not occurred for this "final" keyword, because it is possible to mention the "final" keyword here.

```
Class Test {  
    public static void main(String[] args) {  
        A a = new B();  
        a.m1();  
    }  
}
```

Status: Compilation error.

```

Ex:-
Class A {
    void m1() {
        System.out.println("m1-A");
    }
}

Class B extends A {
    final void m1() {
        System.out.println("m1-B");
    }
}

Class Test {
    public static void main(String[] args) {
        B a = new B();
        a.m1();
    }
}

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

Status:- No compilation Error

Opp:- m1-B.

- ④ If we want to override super class method with sub class method then either super class method or sub class method or both super class and sub class methods must not be declared as static. If we declare either super class or sub class method as static then compiler will raise an error. If we declare both super class and sub class methods as static then compiler will not raise any error but VM will provide super class method output, because super class methods overrides sub class method, they

feature is called as Method Overriding.

~~Error~~

class A

```
Static void m1() {  
    S.O.P("m1-A");  
}
```

Class B extends A {

```
void m1()
```

```
S.O.P("m1-B");  
}
```

here it must be declare the both methods as static, otherwise no one is static.
if we declare both the methods as static, then it gives the output of super class only, because by the static method, superclass method is override sub class methods.

Status:- Compilation error.

$m1()$ in B cannot override $m1()$ in A
 \uparrow
void $m1()$
Overridden method is static.

~~Error~~

class A

```
void m1()
```

```
S.O.P("m1-A");  
}
```

Class B extends A {

```
Static void m1()
```

```
S.O.P("m1-B");  
}
```

Status:- CB

~~Ex/~~ Class A {

```
    static void m1() {
        System.out.println("m1-A");
    }
}
```

Class B extends A {

```
    static void m1() {
        System.out.println("m1-B");
    }
}
```

Class Test {

```
PSVM (String [] args) {
    A a = new B();
}
```

```
    a.m1();
}
```

Status:- No compilation error.

OP:- m1-A .

- (15) If we want to override super class method with sub class method then sub class method must have either same scope or more scope when compared with super class method scope.

Note:- To override super class method with sub class method then sub class method must have either same access privileges or weaker access privileges when compared with super class method access privileges.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

~~Error~~

```
Class A{  
    public void m1() {  
        S.O.P("m1-A");  
    }  
}
```

```
Class B extends A{  
    protected void m1() {  
        S.O.P("m1-B");  
    }  
}
```

Status :- Compilation error.

~~Error~~

```
Class A{  
    protected void m1() {  
        S.O.P("m1-A");  
    }  
}
```

```
Class B extends A{  
    public void m1() {  
        S.O.P("m1-B");  
    }  
}
```

Status :- No Compilation Error.

Q. What are the differences between method Overloading and Method Overriding?

→ ① Method Overloading is a process of extending existed method functionality upto some new functionality.

Method Overriding is the replacement of existed method functionality with new functionality.

② To perform method Overloading, we have to provide different method signature to both the ^{super and sub classes} methods.

To perform method overriding, we must provide the same method prototype for both super class and sub class method.

③ In java apps, we can perform method overloading with or without the Inheritance.

In java apps, we are able to perform method Overriding with only inheritance, it is not possible with out inheritance.

02-10-2015

In java applications, method Overriding is the process of providing replacement for existed method functionality [Super class method] with the new method functionality [Sub class method].

→ In the case of method Overriding, even if we access super class method then you will execute only sub class method functionality, not super class method functionality.

→ In this context, maintaining Super class method implementation is unnecessary, if it is not suggestable.

02-10-2015

→ In the above context, if we want to remove implementation part from a method then we have to declare that method as an abstract method. In java apps if we want to declare any method as an abstract method then the respective class must be abstract class.

→ Abstract method is a java method, it able to have only method declarative part, it will not have method implementation part.

In java applications, abstract methods are allowed in abstract classes and interfaces.

Abstract methods will provide more sharability when compared with concrete methods.

Abstract Class is a java feature, it able to include zero or more no. of concrete methods and zero or more no. of abstract methods.

For abstract classes, we are able to declare reference variables but we are unable to create objects.

To declare abstract classes we have to use special keyword that is "abstract" keyword.

Abstract classes will provide more sharability when compared with concrete classes.

→ In java applications, if we declare any abstract class with abstract methods then it is convention to implement all the abstract methods by taking a sub class.

Z
P.T.O

~~Ex:~~

```

abstract class A {
    void m1() {
        S.O.P("m1-A");
    }
    abstract void m2();
    abstract void m3();
}

class B extends A {
    void m2() {
        S.O.P("m2-B");
    }
    void m3() {
        S.O.P("m3-B");
    }
    void m4() {
        S.O.P("m4-B");
    }
}

class Test {
    public static void main(String[] args) {
        //A a = new A(); → Error
        A a = new B();
        a.m1();
        a.m2();
        a.m3();
    }
}

```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

// A.m4(); → Error

B b = new B();

b.m1();

b.m2();

b.m3();

b.m4();

}

O/P:-	m1-A
	m2-B
	m3-B
	m1-A
	m2-B
	m3-B
	m4-B

→ In java application, if we want to declare a method as an abstract method then the respective class must be an abstract class. If we want to declare a class as an abstract class then it is not at all mandatory condition to declare atleast one abstract method. In java applications, we can declare an abstract class without having abstract methods.

~~Ex:- Class~~

Abstract Class A → If we declared a class as an abstract then it is not at all mandatory condition to declare atleast one abstract method. In java applications, we can declare an abstract class without having abstract methods.

```
Void m1()
{
    S.O.P("M1-A");
}

Void m2()
{
    S.O.P("M2-A");
}

Void m3()
{
    S.O.P("M3-A");
}
```

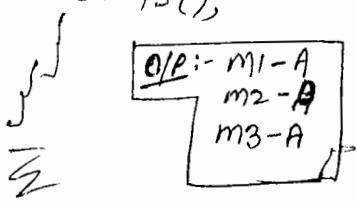
Class B extends A

```
Class Test
{
    PSVM(String[] args)
    {
        .
    }
}
```

//A a=new A();

A a=new B();

```
a.m1();
a.m2();
a.m3();
```



→ In java apps, if we provide implementation for some of the abstract methods in the sub class, then compiler will rise an error. In this case, to come out from compilation error, we have to declare the respective sub class as an abstract class and we have to provide implementation for the remaining abstract methods by taking a sub class through multi level inheritance.

~~Ex:-~~ Abstract Class A

```
Abstract void m1();
Abstract void m2();
Abstract void m3(); }
```

→ Here without declare a class A as an abstract, we can't use the methods which is declared by abstract.

Abstract class B extends A

```
Void m1()
{
    S.O.P("M1-B");
}
```

Class C extends B

```
Void m2()
{
    S.O.P("M2-C");
}

Void m3()
{
    S.O.P("M3-C");
}
```

Class Test

```
Psrvm (String [] args)
{
```

A a = new C();

a.m1();

a.m2();

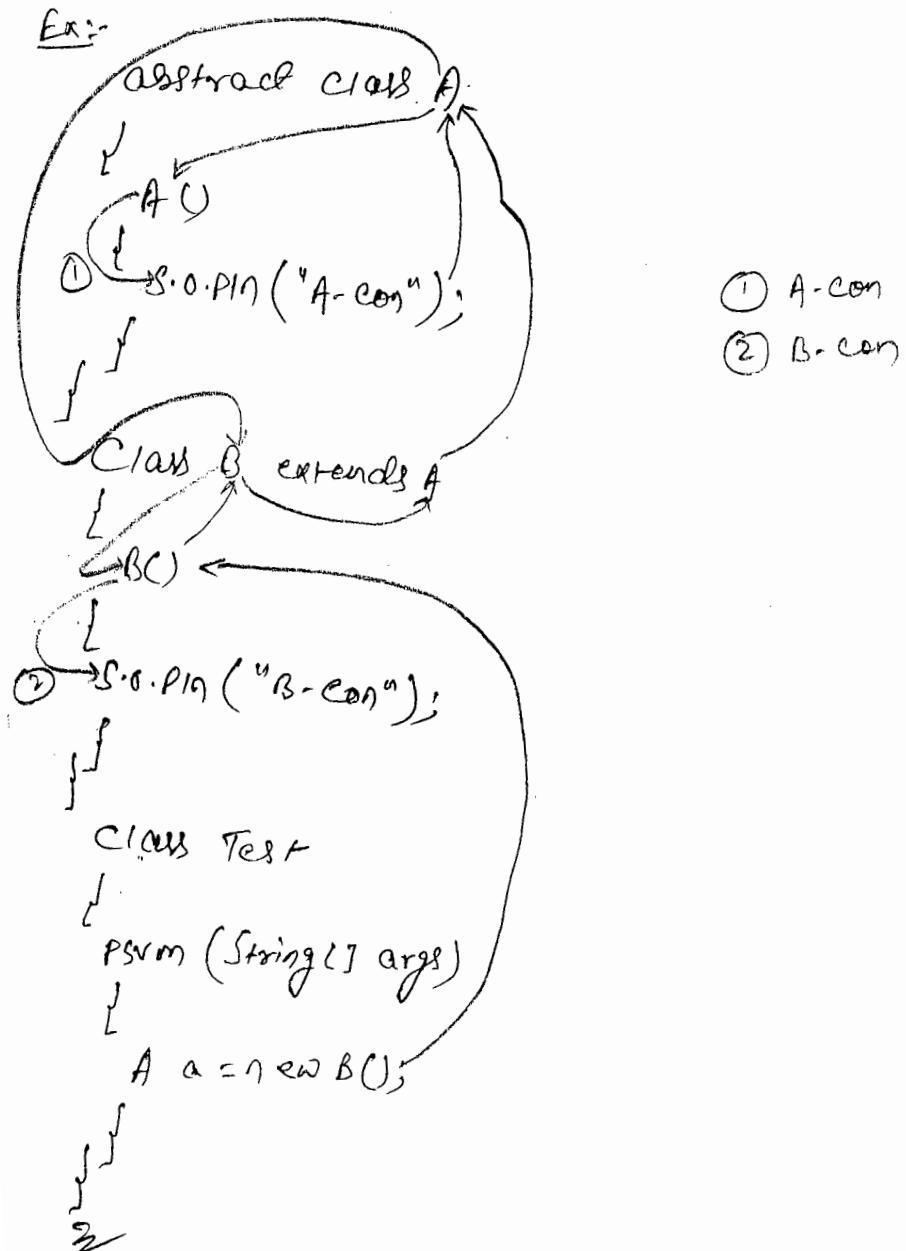
a.m3();

O/P:- M1-B
M2-C
M3-C

03/10/2015

In java applications, we are unable to create object for abstract class but we are able to provide constructors in abstract classes to recognize instance members of the abstract class in order to store in sub class object.

Ex:-



→ In java applications, it is possible to extend an abstract class to concrete class and it is possible to extend a concrete class to an abstract class.

Ex:- Class A → Concrete class.

↳ It can be possible to extend a concrete class to an abstract class.

```

void m1()
{
    System.out.println("m1-A");
}
}

```

Abstract class B → abstract class

```

Abstract void m2();
Abstract void m3();
}

```

Class C extends B → concrete class

↳ It also be possible to extend an abstract class to a concrete class.

```

void m2()
{
    System.out.println("m2-C");
}
void m3()
{
    System.out.println("m3-C");
}
}

```

Class Test

```

public static void main(String[] args)
{
    A a = new C();
    a.m1();
}
}

```

B b = new C();

```

b.m1();
b.m2();
b.m3();
}
}

```

Interfaces:-

- Interface is a java feature, it able to allow Only abstract methods.
 - For interfaces we are able to declare reference variables but we are unable to create objects.
 - In case of interfaces, by default all the variables are "public static final", no need to declare explicitly.
 - In case of interfaces, by default all the methods are "public abstract", no need to declare explicitly.
 - Interfaces are not allowing Constructors, but classes and abstract classes are allowing Constructors.
 - Interfaces will provide more sharability when compared with Classes and abstract classes.
 - To declare interfaces, we have to use special keyword that is "interface".
 - In java apps if we declare any interface with abstract methods then it is convention to implement all the abstract methods by taking an implementation class.
- Ex:-
- ```
interface I {
 int x=10; }
 void m1();
 void m2();
 void m3(); }
```
- Special keyword "interface" using only for interface.  
In the case of interface all the variables are by default "public static final", no need to declare explicitly.  
In the case of interface all the methods are by default "public abstract", no need to declare explicitly.

P.T.O.

~~class A extends I~~

Class A implements I → Class can implement an interface.  
don't extend the interface.

```
public void m1()
```

```
{ S.O.P1("m1-A"); }
```

```
public void m2()
```

```
{ S.O.P1("m2-A"); }
```

```
public void m3()
```

```
{ S.O.P1("m3-A"); }
```

```
public void m4()
```

```
{ S.O.P1("m4-A"); }
```

```
}
```

Class Test

```
{
```

```
public (String[] args)
```

```
// I i = new I(); → Error
```

```
I i = new A();
```

```
S.O.P1(I.x);
```

```
S.O.P1(A.x);
```

```
S.O.P1(i.x);
```

```
i.m1();
```

```
i.m2();
```

```
i.m3();
```

Here, error occurs because we are trying to access the object.

// i.m4(); → Error

A a = new A();

S.O.P1(a.x);

a.m1();

a.m2();

a.m3();

a.m4();

}

Here, error not occurred, because we are taking an object of class A with the ref. variable of class A which is subclass. So, we can call both interface and methods.

→ In java applications, if we declare any implementation class for an interface then it is mandatory to implement all the abstract methods of the respective interface in the implementation class. In this context, if we implement some of the abstract methods of an interface in the implementation class then compiler will rise an error. In this context, to come out from compilation errors we have to declare the respective implementation class as an abstract class and we have to provide implementation for the remaining abstract methods by taking a subclass for the implementation class.

Ex:-

interface I

```
void m1();
void m2();
void m3();
```

```
{
 abstract class A implements I
 {
 public void m1()
 {
 S.O.P("m1-A");
 }
 }
}
```

Class B extends A

```
{
 public void m2()
 {
 S.O.P("m2-B");
 }
}
```

These methods are by default "public abstract" in interface no need to declare explicitly.

→ Here, implementation class A take as abstract class A because if declare three methods in interface I we must implement these methods in implementation class A. If we declare only some methods in implementation class then we must declare these class as an abstract class. otherwise compiler will rise an error.

→ After declaring the implementation class A as abstract class we can declare the remaining methods in the sub class of class A which is class B. (without declaring abstract).

```
public void m3()
{
 System.out.println("m3-B");
}
```

CLASS Test

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

```
 I i = new B();
 i.m1();
 i.m2();
 i.m3();
```

```
}
```

```
}
```

```

Class A implements I1, I2, I3
{
 public void m1()
 {
 System.out.println("m1-A");
 }

 public void m2()
 {
 System.out.println("m2-A");
 }

 public void m3()
 {
 System.out.println("m3-A");
 }
}

```

Here by the class A it's possible to implement more than one interface, by declaring the methods in implementation class which is declared in interface.

```

Class Test
{
 PSVM(String large)
}

```

|    |                                                               |                                                                                                                                                                                              |
|----|---------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| I1 | <code>e1 = new A();<br/>e1.m1();</code>                       | → Here, we can't access all the methods by declaring one object with reference variable of interface, so, we must have to use separate reference variables for separate methods of a Object. |
| I2 | <code>e2 = new A();<br/>e2.m2();</code>                       |                                                                                                                                                                                              |
| I3 | <code>e3 = new A();<br/>e3.m3();</code>                       |                                                                                                                                                                                              |
| A  | <code>a = new A();<br/>a.m1();<br/>a.m2();<br/>a.m3();</code> | → Here, we can access all the methods by using reference variables by implementation class object (without using reference variable for interface.)                                          |

In java applications, it is not possible to extend more than class to a single class, but it is possible to extend more than one interface to a single interface.

Ex:-

```
interface I1
{
 void m1();
}

interface I2
{
 void m2();
}

interface I3 extends I1, I2
```

[Note:- It is not possible to extend more than class to a single class because of it multiple inheritance will be occurred for the class, but not for interface.]

```
Class A implements I3
{
 public void m1()
 {
 System.out.println("m1-A");
 }

 public void m2()
 {
 System.out.println("m2-A");
 }

 public void m3()
 {
 System.out.println("m3-A");
 }
}
```

Here, it is possible to extend more than one interface to a single interface and when we want to extend interface to interface extend keyword is used not used implement keyword.

Class Test

```
{}
public static void main(String[] args)
{
 I1 i1 = new A();
 i1.m1();
 I2 i2 = new A();
 i2.m2();
 I3 i3 = new A();
 i3.m1(); i3.m2(); i3.m3();
}
```

04/10/2015

In java applications, if we want to extend a Super class and implement an interface within a single java class then, first, we have to extend Super class then we have to implement interface. We are unable to interchange "extends" and "implements" keywords in implementation class.

~~Ex:-~~

interface I

```
{ void m1();
void m2(); }
```

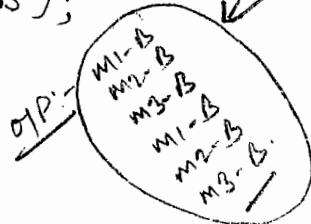
Class A {

```
{ void m3();
}
```

```
S.O.P("m3-A");
```

Class B extends A implements I

```
{ public void m1()
{ S.O.P("m1-B");
}
public void m2()
{ S.O.P("m2-B");
}
```



Some doubt is here  
save by Interfaceoverriding.java in notepad.  
doubt → why it is not printed the  
value of super class by the help  
of reference variable of super  
class.

```
Class Test {
Psvm (String [] args) {
```

```
I i = new B();
```

```
i.m1();
```

```
i.m2();
```

```
// i.m3(); → Error
```

```
A a = new B();
```

```
a.m3();
```

```
// a.m1(); → Error
```

```
// a.m2(); → Error
```

```
B b = new B();
```

```
b.m1(); b.m3(); } }
```

Q Find the valid syntaxes between classes, abstract classes and interfaces from the following list,

- ① Class extends class → Valid ✓
- ② Class extends class, class → Invalid ✓
- ③ Class extends abstract class → Valid ✓
- ④ Class extends abstract class, abstract class → Invalid ✓
- ⑤ Class extends class, abstract class → Invalid ✓
- ⑥ Class extends interface → Invalid ✓
- ⑦ Class implements interface → Valid ✓
- ⑧ Class implements interface, interface → Valid ✓
- ⑨ Class implements interface extends class → Invalid ✓
- ⑩ Class implements interface extends abstract class → Invalid ✓
- ⑪ Class extends class implements interface → Valid ✓
- ⑫ Class extends abstract class implements interface → Valid ✓

- ① abstract class extends class → Valid ✓
- ② abstract class extends class, class → Invalid ✓
- ③ abstract class extends abstract class → Invalid ✓
- ④ abstract class extends abstract class, abstract class → Invalid ✓
- ⑤ abstract class extends abstract class, class → Invalid ✓
- ⑥ abstract class extends interface → Invalid ✓
- ⑦ abstract class implements interface → Valid ✓
- ⑧ abstract class implements interface, interface → Valid ✓
- ⑨ abstract class implements interface extends class → Invalid ✓
- ⑩ abstract class implements interface extends abstract class → Invalid ✓
- ⑪ abstract class extends class implements interface → Valid ✓
- ⑫ abstract class extends abstract class implements interface → Valid ✓

- ① interface extends class → Invalid ✓
  - ② interface extends abstract class → Invalid ✓
  - ③ interface extends interface → Valid ✓
  - ④ interface extends interface, interface → Valid ✓
  - ⑤ interface implements interface → Invalid ✓
- ≡

→ The process of interacting with database from java application is called as JDBC. In JDBC applications we have to use drivers to map java representations with Query language representations and Query language representations with Java representations.

To provide Driver as a product, sun microsystems has provided `java.sql.Driver` as an interface and given an option to all database vendors to provide their own implementation classes to `java.sql.Driver` interface.

→ With the above convention, all the database vendors have provided their own implementation classes in their database softwares.

Ex:-

interface DB-Driver //sun Microsystems

{  
    Void registerDriver();

    Void Connect();

} class Oracle-DB-Driver implements DB-Driver // Oracle

{  
    public Void registerDriver()

        S.O.P("Oracle-DB-Driver is registered");

}

09/10/2015

(156)

public void connect()

{  
S.O.P("Connection is established between java application  
and Oracle DB");  
}  
}

Class MySQL\_DB\_Driver implements DB\_Driver // MySQL

{  
public void registerDriver()  
{

S.O.P("MySQL\_DB\_Driver is registered");  
}

public void connect()  
{

S.O.P("Connection is established between java application  
and MySQL DB");  
}  
}

Class DB2\_DB\_Driver implements DB\_Driver // DB2

{  
public void registerDriver()  
{

S.O.P("DB2\_DB\_Driver is registered");  
}

public void connect()  
{

S.O.P("Connection is established between java application  
and DB2 DB");  
}  
}

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.



```
Class TabcApp
{
 public (String [] args)
 {
 DB-Driver oracle-Driver = new Oracle-DB-Driver();
 Oracle-Driver.registerDriver();
 Oracle-Driver.connect();
 System.out();
 }

 DB-Driver MySQL-Driver = new MySQL-DB-Driver();
 MySQL-Driver.registerDriver();
 MySQL-Driver.connect();
 System.out();

 DB-Driver db2-Driver = new DB2-DB-Driver();
 db2-Driver.registerDriver();
 db2-Driver.connect();
}

}

```

~~Maxima Database~~

## Marker Interface:-

(15)

Marker interface is an interface declared without the abstract methods and providing some capabilities to the objects at runtime.

Ex:-

java.io.Serializable

java.lang.Cloneable

### ① java.io.Serializable:-

The process of separating data from an object is called as Serialization.

The process of getting an object on the basis of the data is called as Deserialization.

In java applications, by default, all the objects are not eligible for serialization and deserialization but the objects which are implementing java.io.Serializable interface are eligible for serialization and deserialization.



### ② java.lang.Cloneable:-

The process of generating duplicate object from an object is called as Object Cloning.

In java applications, by default, all java objects are not eligible for Object Cloning, but, only the objects which are implementing java.lang.Cloneable interface are eligible for Object cloning.



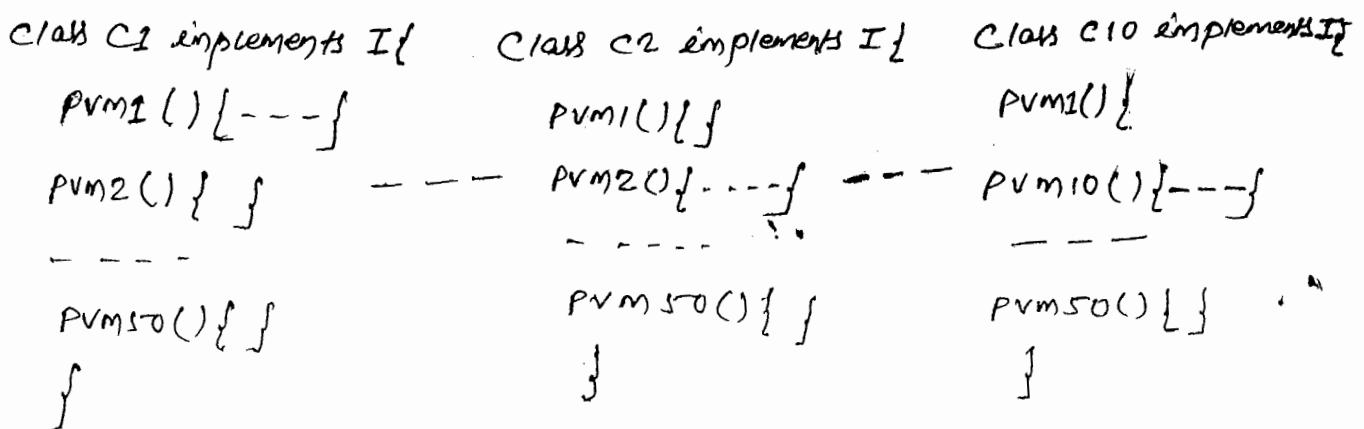
05/10/2015

### Adapter Classes:-

- In java applications, if we declare no. of interfaces with abstract methods and we may implement these interfaces in more no. of implementation classes as per the application requirement.
- In java applications, if we implement any interface in an implementation class then it is mandatory to implement all the abstract methods in the respective implementation class, irrespective of the actual application requirement, this approach will increase no. of unnecessary methods implementation, it is not suggestable in java application development.

```
interface I {
 void m1();

 void m50();
}
```



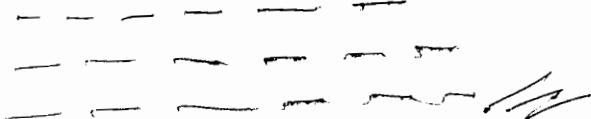
In the above situation to reduce unnecessary methods implementation while implementing interface "Adapter Design pattern" has provided the following solution.

- ① Declare a mediator class between implementation classes and Interface.
- ② Implement interface directly in mediator class and provide empty implementation for each and every method.
- ③ At implementation classes, extends mediator class instead of implementing interface and Override only the required methods instead of implementing all the methods.
- ④ It is suggestable to declare mediator class as an abstract class in order to avoid the option of creating objects for Mediator class.
- ⑤ If we require any method common to all the implementation classes with variable implementation part then it is suggestable to declare that method as an abstract method in the mediator class.

→ In the above solution, the mediator class which we have taken is called as "Adapter Class".

#### Examples:-

- ① WindowAdapter
- ② MouseAdapter
- ③ KeyAdapter
- ④ GenericServlet
- ⑤ ServletRequestWrapper



interface I{

void m1();

-----  
void m50();  
}

abstract class M implements I{

PvM1(){ };

-----  
abstract PvM25();

PvM50(){ };

Adaptor Class or  
Generic Class.

Class C1 implements I {  
extends M

PvM1(){---};  
PvM25(){---};  
}  
-----

Class C2 implements I {  
extends M

PvM2(){---};  
PvM25(){---};  
}

Class C10 implements I {  
extends M

PvM10(){---};  
----- PvM25C(){---};  
}

WindowListener

P v windowOpened (WE we)  
P v windowClosed (WE we)  
P v windowClosing (WE we)  
P v windowActivated (WE we)  
P v windowDeactivated (WE we)  
P v windowIconified (WE we)  
P v windowDeiconified (WE we)

WindowAdapter

P v windowOpened (WE we){ }  
P v windowClosed (WE we){ }  
P v windowClosing (WE we){ }  
P v windowActivated (WE we){ }  
P v windowDeactivated (WE we){ }  
P v windowIconified (WE we){ }  
P v windowDeiconified (WE we){ }

WindowListener

P v windowClosing (WindowEvent we){  
System.exit(0);  
}

extends

JAVA 8 features over interfaces:- (Not for industry uses) → Obtated

153

- ① Static methods in interfaces
  - ② Default methods in interfaces
  - ③ Functional Interfaces

### ① Static methods in interfaces :-

Upto JAVA7 version we are able to provide Only abstract methods inside the interfaces, but JAVA8 version has provided a flexibility to declare static methods inside the interfaces in order to improve Sharability.

If we declare static method inside interface then we are able to access that static method by using Only respective interface name directly, it is not possible to access that static method by using interface reference variable, implementation class name and implementation class reference Variable.

```

interface I
{
 static void m1()
 {
 System.out.println("m1-A");
 }
}

class A implements I
{
}

class Test {
 public static void main(String[] args)
 {
 I.m1();
 /* A a=new A();
 * a.m1(); → Error
 */
 }
}

```

06/10/2015

## (2) Default Methods:-

In general, in interfaces we will declare some service names in the form of abstract methods and these services are implemented by some other developer or some other team or some third party ~~vendors~~ vendors. In this context, JAVA has provided a flexibility for the developers like to provide initial implementation or default implementation for the interface methods or services.

If we provide default implementation for interface methods inside the interface that we may or may not override in the implementation class as per the requirement.  
To declare default methods inside the interface, we have to use "default" keyword.

Ex:-

```
interface Course
{
 default String getId()
 {
 return "C-111";
 }

 default String getName()
 {
 return "JAVA";
 }

 default int getCost()
 {
 return 5000;
 }
}
```

Class CourseImpl1 implements Course

```
{
 public String getCid()
 {
 return "C-222";
 }
 public String getName()
 {
 return ".NET";
 }
}
```

Class CourseImpl2 implements Course

```
{
 public String getCid()
 {
 return "C-333";
 }
 public String getCost()
 {
 return 10000;
 }
}
```

Class Test

```
{
 public static void main(String[] args)
 {
 Course c1 = new CourseImpl1();
 System.out.println("Course Details");
 System.out.println("-----");
 System.out.println("Course Id :" + c1.getCid());
 System.out.println("Course Name :" + c1.getName());
 System.out.println("Course Cost :" + c1.getCost());
 System.out.println();
 }
}
```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

```

Course c2 = new CourseImpl2();
System.out.println("Course Details");
System.out.println("-----");
System.out.println("Course Id :" + c2.getId());
System.out.println("Course Name :" + c2.getName());
System.out.println("Course Cost :" + c2.getCost());
}
}

```

### @Functional Interface:-

If any interface allows exactly one abstract method then that interface is called as functional interface.

→ In case of functional interfaces we are unable to provide more than one abstract methods and 0% of abstract methods, but we are able to provide more than one default methods and static methods inside the functional interfaces.

Ex:- java.lang.Runnable interface is functional interface.

Ex:-  
@FunctionalInterface  
interface I

```
{
 void m1();
}
```

```
// void m2(); → Error
```

Static void m2() and (0%)

```
{
 System.out.println("m2 - I");
}
```

Here, in functional interface we cannot use more than one abstract interface, but we can use more than one default methods and static methods inside the functional interfaces.

```

 static void m3()
 {
 System.out("m3-I");
 }

 default void m4()
 {
 System.out("m4-I");
 }

 default void m5()
 {
 System.out("m5-I");
 }
}

```

```

class A implements I
{
 public void m1()
 {
 System.out("m1-A");
 }
}

class Test
{
 public static void main(String[] args)
 {
 A i = new A();
 i.m1();
 i.m2();
 i.m3();
 i.m4();
 i.m5();
 }
}

```

$i \cdot m_3()$  → Here, reference variable  $i$  cannot access static method  
 $i \cdot m_4()$  so, we have to access by  $I \cdot$   
 $i \cdot m_5()$  → The default methods are always accessed by reference  
variable ( $i$ ) not by others.

## Object Cloning:-

The process of generating a duplicate object from an object is called as Object cloning.

- In java apps, by default all the objects are not eligible for cloning, but ~~the~~ only the objects which are implementing `java.lang.Cloneable` marker interface are eligible for Object cloning.
- If we want to perform Object Cloning then we have to use the following steps.

① Declare a class by implementing `java.lang.Cloneable` interface.

```
Class MyClass implements Cloneable
{
}
}
```

② Override `Object` class `clone()` method in user defined Class

```
Class MyClass implements Cloneable
{
 public Object clone() throws CloneNotSupportedException
 {
 return super.clone();
 }
}
```

③ Access clone() method to get Duplicate Object in main() method.

MyClass me1 = new MyClass();

MyClass me2 = (MyClass) me1.clone();

Eg:-

Class Employee implements Cloneable

```

 {
 String eid;
 String ename;
 float esal;
 String eaddr;
 }
```

Employee( String eid, String ename, float esal, String eaddr)

```

 {
 this.eid = eid;
 this.ename = ename;
 this.esal = esal;
 this.eaddr = eaddr;
 }
```

public void getEmpDetails()

```

 {
 System.out.println("Employee Details");
 System.out.println("-----");
 System.out.println("Employee Id :" + eid);
 System.out.println("Employee Name :" + ename);
 System.out.println("Employee Salary :" + esal);
 System.out.println("Employee Address :" + eaddr);
 }
```

public Object clone() throws CloneNotSupportedException

```

 {
 return super.clone();
 }
}
```

Class Test

{

psvm (String [] args) throws Exception

{

Employee emp1 = new Employee("E-111", "AAA", 5000, "Hyd");

s.o.pn ("Employee details from Original Object")

emp1.getEmpDetails();

Employee emp2 = (Employee) emp1.clone();

s.o.pn();

s.o.pn ("Employee details from Duplicate Object");

emp2.getEmpDetails();

}

3

There are two types of cloning in java.

- ① Shallow Cloning
- ② Deep Cloning.

### ① Shallow Cloning:-

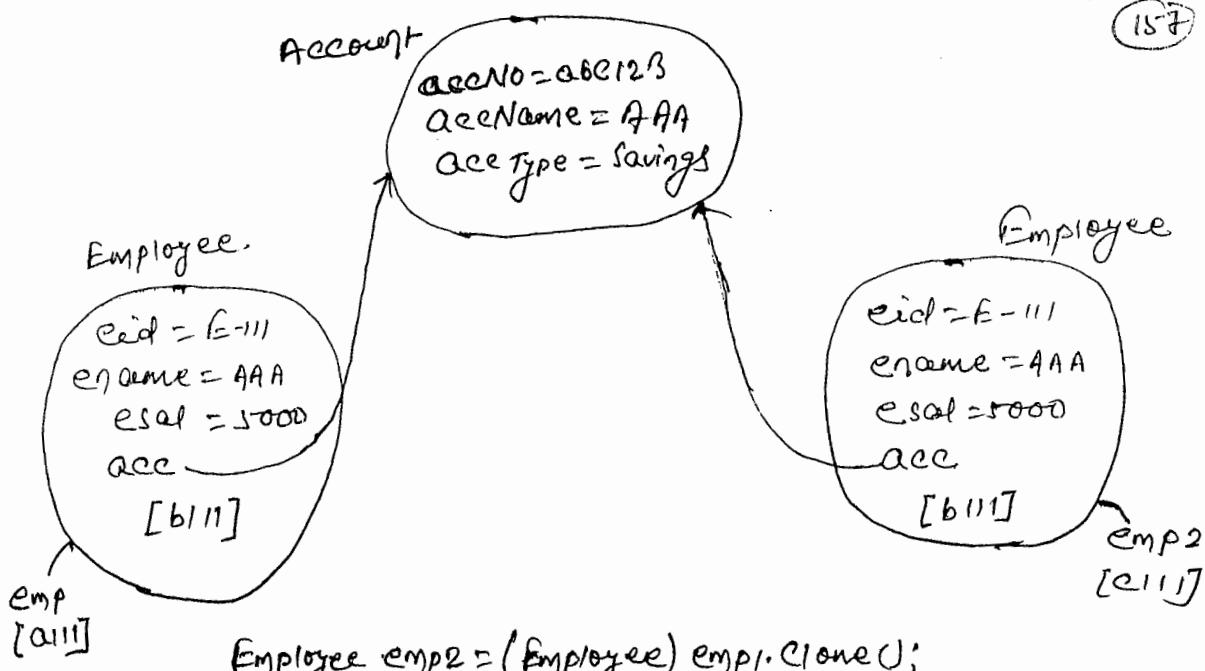
This cloning mechanism is default cloning mechanism in java apps.

→ In case of shallow cloning, JVM will perform cloning over only Container Objects, not over Contained Objects if any association exists. In this context, both Original Container Object and Duplicate Container object will refer the same Contained Object.

3

07/10/2015

(157)



Ex:-

```

Class Account {
 String accNo;
 String accName;
 float esal;
 String accType;
 Account(String accNo, String accName, float esal, String accType)
 {
 this.accNo = accNo;
 this.accName = accName;
 this.esal = esal;
 this.accType = accType;
 }
}

```

Class Employee implements Cloneable

```

 {
 String eid;
 String ename;
 float esal;
 Account acc;
 Employee(String eid, String ename, float esal, Account acc)
 }

```

```

 {
 this.eid = eid;
 this.ename = ename;
 this.esal = esal;
 this.acc = acc;
 }

 public void getEmpDetails()
 {
 System.out.println("Employee Details");
 System.out.println("-----");
 System.out.println("Employee Id :" + eid);
 System.out.println("Employee Name :" + ename);
 System.out.println("Employee Salary :" + esal);
 System.out.println("Account Details");
 System.out.println("-----");
 System.out.println("Account Number :" + acc.accNo);
 System.out.println("Account Name :" + acc.accName);
 System.out.println("Account Type :" + acc.accType);
 }

 public Object clone() throws CloneNotSupportedException
 {
 return super.clone();
 }
}

Class Test {
 public void main(String[] args) throws Exception
 {
 Account acc = new Account("Abc123", "AAA", "Savings");
 Employee emp = new Employee("E-11", "AAA", 5000, acc);
 }
}

```

S.O.P19 ("Employee Details From Original Object");

empl.getEmpDetails();

S.O.P19 ("Original Emp Ref :" + emp1);

S.O.P19 ("Original Acc Ref :" + acc);

Employee emp2 = (Employee) emp1.clone();

S.O.P19();

S.O.P19 ("Employee Details from Duplicate Object");

emp2.getEmpDetails();

S.O.P19 ("Duplicate Emp Ref :" + emp2);

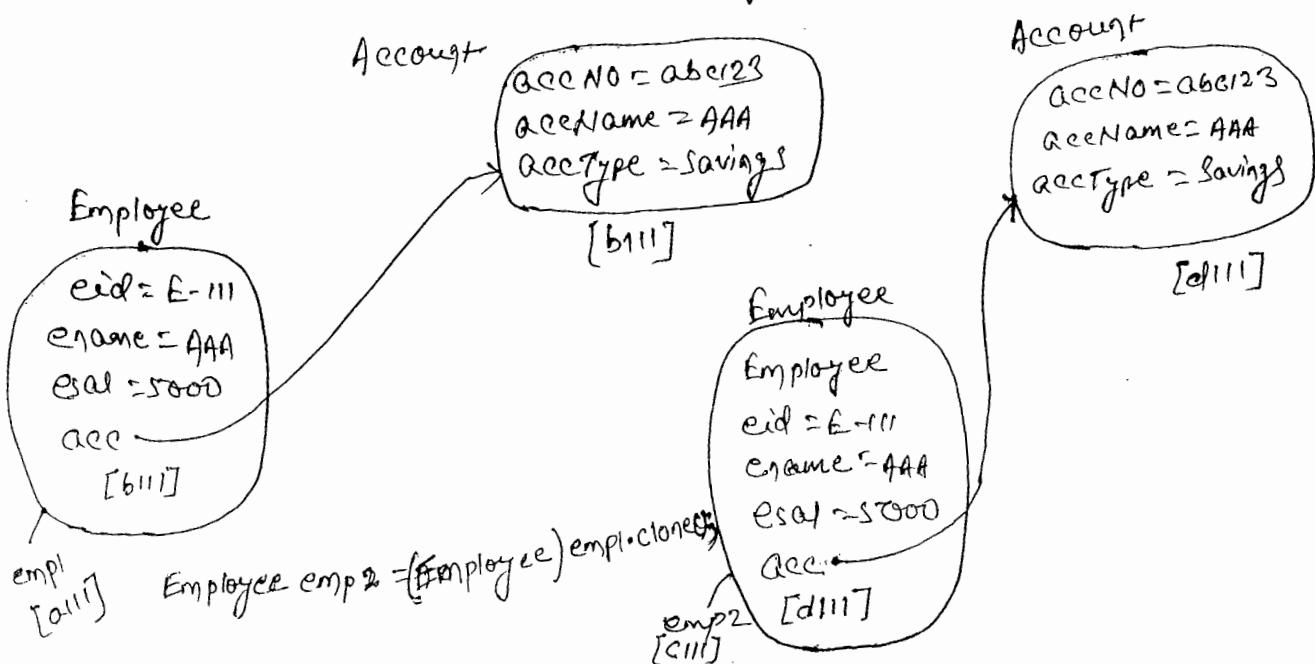
S.O.P19 ("Duplicate Acc Ref :" + emp2.acc);

}

## ② Deep Cloning:-

Deep Cloning is not default Cloning mechanism in java applications.

→ In case of deep cloning, JVM will perform Cloning over Container Objects as well as contained Objects if any association is existed.



To get deep Cloning.

In shallow cloning program, implement clone() method in Employee class like below:

public Object clone() throws CloneNotSupportedException

```
 Account acc = new Account(accno, accname, acc.
 accType);
```

```
Employee emp = new Employee(eid, ename, esal, acc);
```

```
 return emp;
```

}

### instance of Operator:-

'instanceof' Operator is Boolean Operator, it will check whether the specified reference variable is representing an instance of the specified class or not.

#### Syntax:-

ref-var instanceof Class-name

- ① If ref-var type is not derived with the Specified Class-name through inheritance then Compiler will raise an error.
- ② If ref-var type is same as the Specified class name then instanceof Operator will return true value.
- ③ If ref-var type is sub class to the Specified class name then instanceof Operator will return true value.

⑤ If ref-var is super class to the specified class name  
then instanceof ~~is~~ operator will return false value.

Ex:-

```
Class A
{
}
```

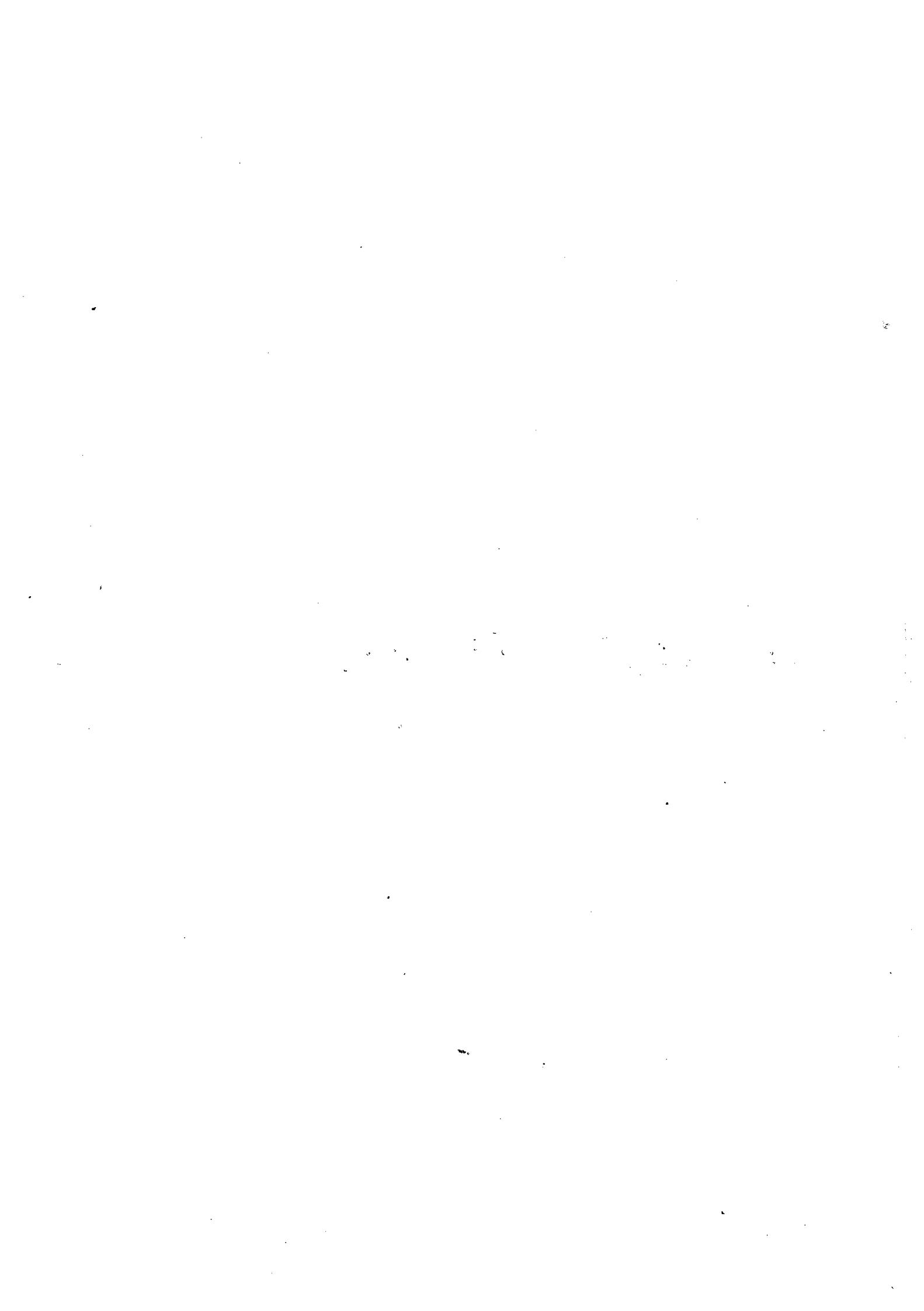
```
Class B extends A
{
}
```

```
Class C
{
}
```

```
Class Test {
 PSVM (String [] args) {
 A a = new A();
 S.O.Pn (a instanceof A);
 B b = new B();
 S.O.Pn (b instanceof B);
 S.O.Pn (b instanceof A);
 S.O.Pn (a instanceof B);
 C c = new C();
 S.O.Pn (c instanceof C);
 //S.O.Pn (c instanceof A); → Error
 }
}
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.





### Reflection API:-

The process of analysing all the capabilities of a particular class is called as "Reflection".

In reflection, we will read metadata of the programming elements like variables, methods, constructors, classes, ..... Where metadata includes access modifiers list, names, super class details, implemented interfaces details, return type, data types, parameter list, ...

To read the above specified metadata of the programming elements JAVA has provided a set of predefined classes called as "Reflection API".

Reflection API is ~~very~~ <sup>not</sup> useful in service Oriented projects development, Reflection API is very much usefull in products development like Compilers, JVMs, Servers, frameworks, Testing tools, debugging tools, ... ,

Ex:- ① In java applications, to provide compilation errors, Compiler must check all the signatures of the programming elements, for this Compiler must read metadata of the programming elements. To achieve this requirement, Compilers must use Reflection API internally.

Ex:- ② \*

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

JAVA has provided the complete predefined library to perform reflection in the form of `java.lang.reflect` package.

→ Reflection API has included the following classes to represent metadata of the programming elements like variables, methods, Constructors, classes, ...

`java.lang.Class`

`java.lang.reflect.Field`

`java.lang.reflect.Constructor`

`java.lang.reflect.Method`

+ - - - -

- - - - -

`java.lang.Class` :-

This class object is able to represent metadata of the particular class including access modifiers list, class name, super class metadata, implemented interfaces metadata, declared variables metadata, declared methods metadata, declared constructors metadata, ...

There are three ways to get `java.lang.Class` object in order to represent classes metadata.

- ① By using `Class.forName()`
- ② By using `getClass()` method from Object class
- ③ By using "Class" static variable.

① By using `Class.forName()`

`public static Class forName(String class-name) throws java.lang.  
ClassNotFoundException`

Ex:- `Class C = Class.forName("Employee");`

When JVM encounters the above instruction, JVM will perform the following actions.

- ① JVM will take the provided class name from fname() method.
- ② JVM will search for the respective .class file at current location, at java predefined library and at the locations of the "Classpath" environment variable.
- ③ If the required .class file is not available at all the above specified locations then JVM will raise an exception like `java.lang.ClassNotFoundException`
- ④ If the required .class file is available at ~~the~~ either of the above locations then JVM will load its bytecode to the memory.
- ⑤ After loading class bytecode to the memory, JVM will get metadata of the loaded class and JVM will store that metadata in the form of `java.lang.Class` object in heap memory.

## ② By Using `getCLASS()` method from `Object Class`:

```
public class getClass()
```

Ex:- Employee emp = new Employee();

Class C = emp.getClass();

In java applications, for each and every class loading JVM <sup>will</sup> create a separate `java.lang.Class` object in heap memory, if includes metadata of the loaded class.

In the above example, when we create Object for Employee class then JVM will load Employee class bytecode to the memory and JVM will create `java.lang.Class` object with Employee class metadata in heap memory. In this context, we can use `getClass()`

Method of `java.lang.Object` class to get the generated Class object.

③ By using "Class" static variable:-

`public static final Class class;`

In java applications, when we compile a class, compiler will append a "Class" static variable to represent `java.lang.Class` object.

Ex:- `Class c = Employee.class;`

Ex:-

```
public class Employee implements Serializable, Cloneable
{
 public String eid;
 public String ename;
 public String eaddr;
 public Employee(String eid)
 {
 }
 public Employee(String eid, String ename)
 {
 }
 public Employee(String eid, String ename, String eaddr)
 {
 }
 public void add(String eid, String ename, String eaddr)
 {
 }
```

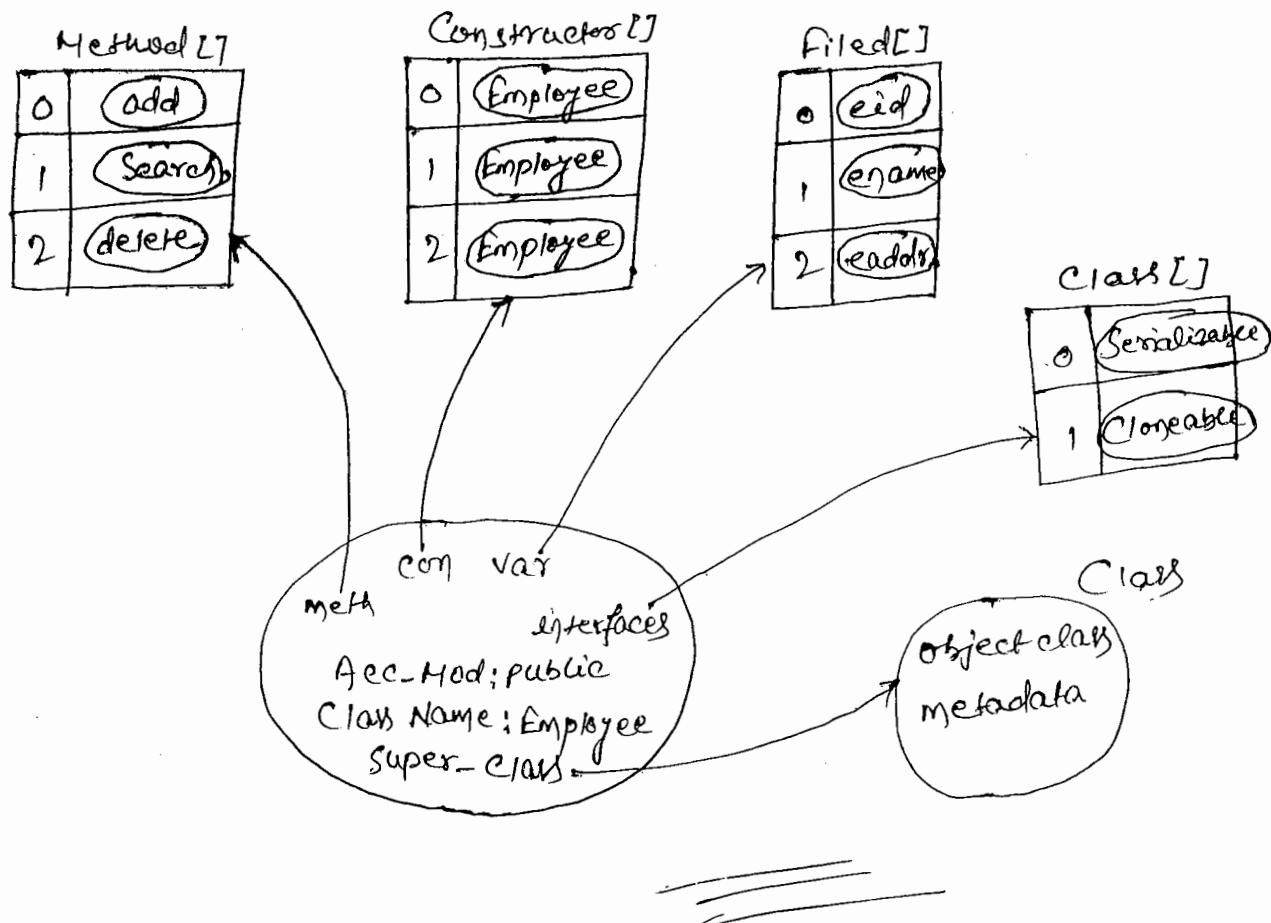
```

public void search (String eid)
{
}
}

public void delete (String eid)
{
}
}

```

Class c = class.forName("Employee");



P.T.O

To get class name from `java.lang.Class` object we have to use the following method.

```
public String getName()
```

To get super class metadata in the form of `java.lang.Class` Object we have to use the following method.

```
public Class getSuperclass()
```

To get implemented interfaces list in the form of `Class[]` we have to use the following method.

```
public Class[] getInterfaces()
```

→ To get access modifiers of the class, we have to use the following methods.

```
public int getModifiers() from Class
```

```
public static String toString(int val) from java.lang.reflect.Modifier
```

Class.

Ex:- `int val = c.getModifiers();`

```
String modList = Modifier.toString(val);
```

Ex:-

```
import java.lang.reflect.*;
public abstract class Employee implements java.io.Serializable,
Cloneable
```

{

}

```
class ClassA
```

{

```
public void main(String[] args) throws Exception
```

{

```
Class c = Class.forName("Employee");
```

```
System.out.println("Class Name : " + c.getName());
```

```

S.O.P("Super Class :" + c.getSuperclass().getName()); 163
Class[] c1s = c.getInterfaces();
S.O.P("Interfaces :");
for(Class c1: c1s)
{
 S.O.P(c1.getName() + " ");
}
S.O.P();
int val = c.getModifiers();
String mod_list = Modifiers.toString(val);
S.O.P("Access Modifiers :" + mod_list);
}

```

≡

### java.lang.reflect.Field:-

java.lang.reflect.Field class objects are able to store metadata of the variables declare in a class including variable name, variable data type, variable value and access modifiers.

If we want to get metadata of the variables of a particular class, first we have to get java.lang.Class object then we have to get variables metadata in the form of Field[] by using the following methods.

public Field[] getFields()

→ It will include metadata of the public variables defined in the respective class as well as in super class.

P.T.O

public Field[] getDeclaredFields()

→ It will include metadata of the variables which are declared in the respective class, not from super class irrespective of the public declaration.

To get metadata of the variables like variable name, variable data type, variable value and access modifiers of the variable we have to use the following methods from `java.lang.reflect.Field` class

```
public String getName()
public Class getType()
public xxx getField(f)
Where, xxx may be byte, short, int, ...
public int getModifiers()
```

Ex:-

```
import java.lang.reflect.*;
class Employee
{
 public static String eid = "E-111";
 static final String ename = "Durga";
 protected static final float esal = 5000.0f;
}
class FieldEx
{
 public void main(String[] args) throws Exception
 {
 Employee emp = new Employee();
```

```

Class C = emp.getClass();
Field[] fields = C.getDeclaredFields();
for (Field f : fields)
{
 System.out("VarName :" + f.getName());
 System.out("Data type :" + f.getType().getName());
 System.out("Value :" + f.get(f));
 String modList = Modifier.toString(f.getModifiers());
 System.out("Acc-Mod :" + modList);
}
}

```

### java.lang.reflect.Method:-

java.lang.reflect.Method class object is able to store metadata of a particular method.

If we want to get methods metadata of a particular class, first we have to create Class object then we have to use following methods.

public Method[] getMethods()

→ It will get Only public methods metadata from the respective class and from the Super class.

public Method[] getDeclaredMethods()

→ It will get all the methods metadata which are declared by the respective class, not from Super class.

To get metadata of a method like method name, return type, access modifiers list, parameter types, exception types then we have to use the following methods.

```
public String getName()
public Class getReturnType()
public Class[] getParameterTypes()
public Class[] getExceptionTypes()
public int getModifiers()
```

Ex:-

```
import java.lang.reflect.*;
class Employee
{
 public static void add (String eid, String ename, float esal,
 String eaddr) throws ArithmeticException, NullPointerException
 {
 public final void search (String eid, String ename) throws
ArrayIndexOutOfBoundsException, ArrayIndexOutOfBoundsException, ClassCastException
 {
 public synchronized void delete (String eid) throws
 java.rmi.RemoteException, java.sql.SQLException
 {
 }
 }
 }
 }
```

*Method* → *Constructor*

psvm (String[] args) throws exception

```

Class C = Employee::class;
for (Method m: C.getDeclaredMethods())
 for (Method m: m)
 System.out.println("Name :" + m.getName());
 System.out.println("Return Type :" + m.getReturnType() + m.getName());
 int val = m.getModifiers();
 String modList = Modifier.toString(val);
 System.out.println("Access Mod :" + modList);
 Class[] C1 = m.getParameterTypes();
 System.out.println("Param Types :");
 for (Class c1: C1)
 System.out.println(c1.getName() + " ");
 System.out.println();
 Class[] C2 = m.getExceptionTypes();
 System.out.println("Exc Types :");
 for (Class c1: C2)
 System.out.println(c1.getName() + " ");
 System.out.println();
 System.out.println("-----");
 }
}

```

## java.lang.reflect.Constructor:-

This class object is able to ~~not~~ store metadata of a particular constructor.

If we want to get metadata of all the constructors of a class, first we have to create java.lang.Class object then we have to use the methods.

public Constructor[] getConstructors()

→ It will return metadata of only public constructors of the class.

public Constructor[] getDeclaredConstructors()

→ It will return metadata of all the constructors irrespective of public declaration.

To get metadata of a constructor like constructor name, constructor access modifiers, parameter types, exception types, ... We have to use the following methods.

public String getName()

public Class[] getParameterTypes()

public Class[] getExceptionTypes()

public int getModifiers()

Ex:-

```
import java.lang.reflect.*;
```

```
class Employee
```

```
{
```

```
 public static void add(String eid, String ename, float esal,
 String eaddr) throws ArithmeticException, NullPointerException
```

```
{
```

```
}
```

→ Please, check one more time  
with others notebook there  
is anything wrong or not.

P.T.O.

(1)

(166)

```
public final void search (String eid, String ename) throws
 ArrayIndexOutOfBoundsException, ClassCastException
{
}
}

public synchronized void delete (String eid) throws
 java.rmi.RemoteException, java.sql.SQLException
{
}

Class ConstructorEx
{
 public (String[] args)
}

Class C = Employee.class;
Constructor[] Con = C.getDeclaredConstructors();
for (Constructor Cn : Con)
{
 System.out.println ("Name : " + Cn.getName());
 System.out.println ("Access Mod : " + Cn.getModifiers());
 String ModList = Modifier.toString (Cn.getModifiers());
 System.out.println ("Access Mod : " + ModList);
 Class[] Cls = Cn.getParameterTypes();
 System.out.println ("param Types : ");
 for (Class Cl : Cls)
 {
 System.out.println (Cl.getName() + " ");
 }
 System.out.println ();
 Class[] Cls2 = Cn.getExceptionTypes();
 System.out.println ("Exc Types : ");
 for (Class Cl : Cls2)
```

```
 }
 S·O·P(C1.getName() + " ");
}
S·O·P(" - - - - - ");
}
=====
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

10/10/2015

(167)

## Wrapper Classes:-

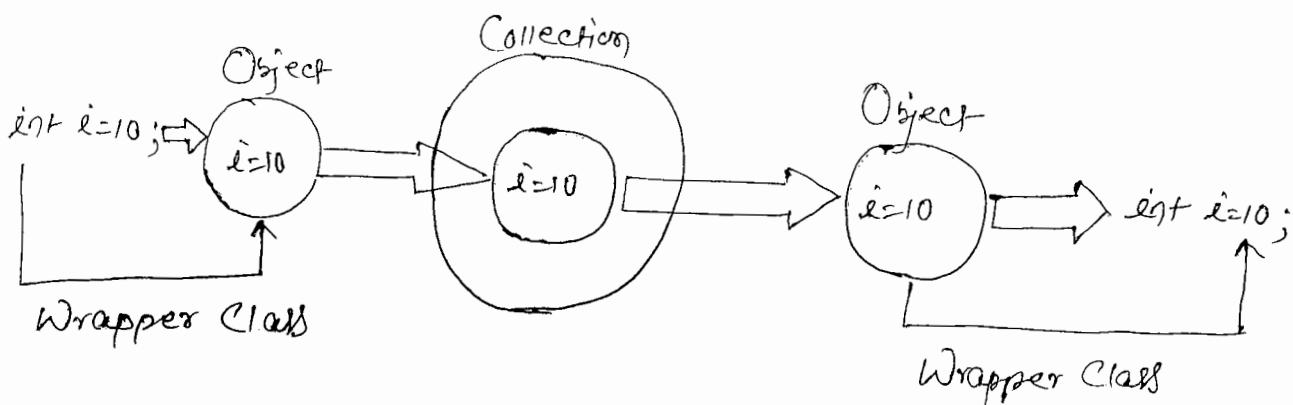
Collection is an Object, it able to store a group of other Objects.

Ex:-  
java.util.ArrayList  
java.util.LinkedList  
java.util.HashSet

In java applications, Collection Objects are able to store only reference values of the objects, not primitive data.

As per the requirement, if you want to manage primitive data in Collection objects, then, first we have to convert data from primitive data type to object type then we have to store object in Collection. If we want to retrieve primitive data from Collection objects, then first we have to retrieve Object from Collection, then we have to convert data from Object type to primitive type.

In the above Context, to Convert data from primitive type to Object type and to convert data from Object type to primitive type JAVA has provided a set of predefined classes called as "Wrapper Classes".



java has provided all the wrapper classes in `java.lang` package w.r.t primitive data types.

### Primitive DTS

byte

short

int

long

float

double

char

boolean

Wrapper Classes

`java.lang.Byte`

`java.lang.Short`

`java.lang.Integer`

`java.lang.Long`

`java.lang.Float`

`java.lang.Double`

`java.lang.Character`

`java.lang.Boolean`

In java all the wrapper class objects are immutable object by nature, they will not allow modifications on their content directly, data is allowed for operations but the resultant data will not be stored back in original object, result data will be stored by creating new object.

### ① Conversions from Primitive type to Object type:-

ⓐ By using parameterized constructor from all the wrapper classes:

`public XXX(XXX value)`

`XXX → Wrapper Classes`

`XXX → Primitive type`

~~XXXX XAKRJW~~

Ex:- `int i=10;`  
`Integer in=new Integer(i);`  
`System.out.println(i+" "+in);`  
O/P:- 10 10

(168)

(b) By Using valueOf(-) method:-

public static XXX valuesOf(XXX value)  
XXX → Wrapper Classes  
XXX → Primitive Types.

Ex:-

`int i=10;`  
`Integer in=Integer.valueOf(i);`  
`System.out.println(i+" "+in);`  
O/P:- 10 10.

(c) By Using Auto-Boxing mechanism:-

→ In this mechanism, no need to use wrapper Class Constructors, predefined methods, ...

Simply we have to assign primitive variable to wrapper class reference variable.

→ This mechanism was provided by JAVA along with its JDK 5.0 version.

Ex:- `int i=10;`  
`Integer in=i;`  
`System.out.println(i+" "+in);`

O/P:- 10 10

## (2) Conversions from Object type to Primitive type :-

### (a) By using xxxValue() method :-

public xxx xxxValue()

xxx → Primitive Type.

Ex:-

Integer i1 = new Integer(10);

i1 + i = i1.intValue();

S.O.P(i1 + " " + i); → S.O.P(i1.intValue() + i);

}

Output:- 10 10

### (b) By Using Auto-Unboxing:-

→ In this mechanism, no need to use any predefined method to convert data from Wrapper Class type to primitive type. Simply, assign wrapper class reference variable to primitive variable.

This mechanism was provided by JAVA along with its JDK5.0 version.

Ex:-

Integer i1 = new Integer(10);

i1 + i = i1;

S.O.P(i1 + " " + i);

}

Output:- 10 10

≡

### ③ Conversion from String type to Object type:-

#### ① By using String parameterized Constructor:-

public xxx(String data)  
where xxx may be byte, short, integer.

Ex:- String data = "10";

Integer in = new Integer(data);

S.O.Pln(data + " " + in);

O/P:- 10 10

#### ② By using valueOf() method:-

public static xxx valueOf(String data)

xxx → Byte, short, integer

Ex:-

String data = "10";

Integer in = Integer.valueOf(data);

S.O.Pln(data + " " + in);

O/P:- 10 10

### ④ Conversions from Object type to String type:-

#### ① By using toString() method:-

public String toString();

Ex:- Integer in = new Integer(10);

String data = in.toString();

S.O.Pln(in + " " + data);

O/P:- 10 10

#### ② By using '+' Concatenation Operator:-

If we concatenate any reference variable with " " then JVM will execute `toString()` method on the specified reference variable internally and it will concatenate the ~~return~~ return value of `toString()` method to " ".

Ex:- Integer in = new Integer(10);  
String data = " " + in;  
System.out.println(in + " " + data);  
O/P:- 10 10

### (5) Conversions from Primitive type to String type:-

- (a) By using static `toString()` from Wrapper Classes:-  
 $\text{xxx} \rightarrow \text{byte, int, short}$ .

Ex:- int i=10;  
String data = Integer.toString(i);  
System.out.println(i + " " + data);  
O/P:- 10 10

- (b) By using "+" Concatenation Operator:-

If we concatenate any primitive variable with " " then JVM will append the value of the specified variable to " ", the overall resultant data is converted as String type.

Ex:- int i=10;  
String data = " " + i;  
System.out.println(i + " " + data);  
O/P:- 10 10

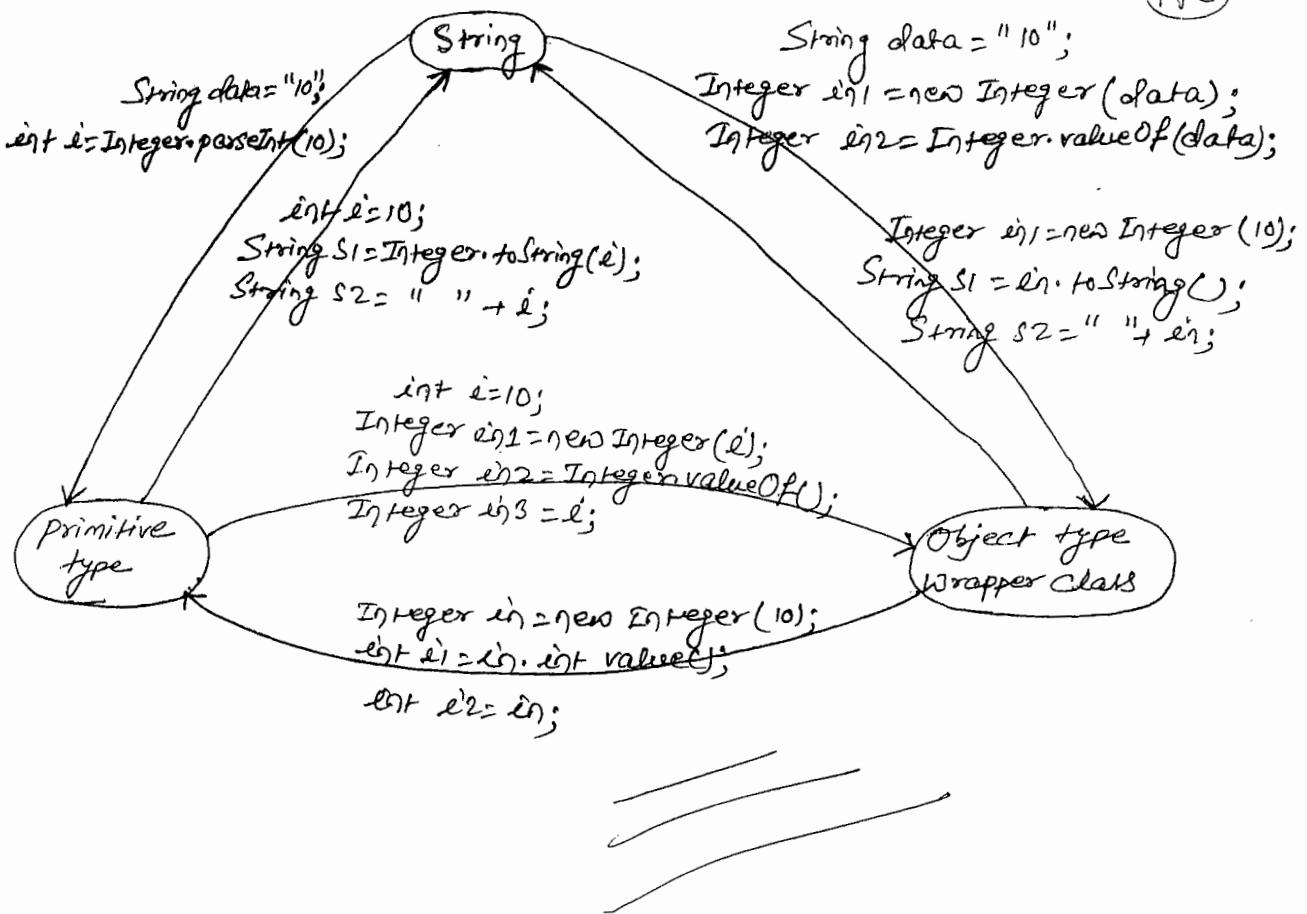
### (6) Conversions from String type to Primitive type:-

- (a) By using `parseXXX()` method from wrapper classes.

public static `XXX parseXXX(String data)` throws NumberFormatException

Ex:- String data = "10";  
int i = Integer.parseInt(data);  
System.out.println(data + " " + i);  
O/P:- 10 10





### SRI RAGHAVENDRA XEROX

Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

## Annotations:-

It's a new feature provided by JDK5.0 version, it can be used to describe metadata in java app.

- Q To provide metadata in java app we are able to use comments then what is the requirement to use "Annotations"?
- In java app, to describe metadata if we use comments then lexical analyzer will remove comments and its metadata from java program's as part of compilation, so that, if we provide metadata along with comments then that metadata is available upto the java files, not available upto class file and upto RUNTIME of our java app.
- As per the requirement, if we want to bring metadata upto java file, upto class files and upto RUNTIME of our java app we have to use "Annotations".

Note:- To simplify debugging, testing --- process we need metadata at RUNTIME of our application. In distributed app's, servers need metadata at runtime of ~~our java app~~ to identify server side components in order to execute server side component.

- Q To provide metadata at runtime of our java app we are able to use XML documents then what is the requirement to use "Annotations".
- To provide metadata at RUNTIME of ~~our~~ java app, ~~that we~~ ~~use~~ XML documents then we are able to get the full drawbacks,

- ① We have to learn XML technology explicitly.
- ② Every time we have to check whether XML documents are located properly or not.
- Every time we have to check whether XML documents are formatted properly or not.

③ Every time we have to check whether we are right passing mechanism or not to read data from XML documents.

To overcome all the above problem we need a java alternative that is "Annotations".

Note:- In java/J2EE application, we are able to use annotations as an alternatives for XML Documents.

### XML Based Technology.

upto jdk 1.4 Ver  
jboss 3.0

Servlets 2.5~

Struts 1.x

JSF 1.x

EJBs 2.x

Spring 2.x

- - - - -

- - - - -

### Annotation Based Technology.

jdk 5.0 and above

JDBC 4.0

Servlets 3.0

Struts 2.x

JSF 2.x

EJBs 3.x

Spring 3.x

- - - - -

- - - - -

In web app to configure servlets in web.xml file we have to use the full XML tags.

### Web.xml

<Web-app>

<Servlet>

<Servlet-name>/></Servlet-name>

<Servlet-class>LoginServlet</Servlet-class>

</Servlet>

<Servlet-mapping>

<Servlet-name>/></Servlet-name>

<Url-pattern>/login</url-pattern>

</Servlet-mapping>

</Web-app>

For all the above XML tags we are able to use full annotations as an alternative in web application.

```
@ web service("/login")
public class LoginService extends HttpServlet
{
}
```

To execute "Annotations" java has provided a separate tool called as "Annotation processing tool" [API], but it was managed by java upto java7 version, in java8 version API was removed from java.

→ If we want to use Annotations in java app then we have to use the full syntax.

declarative Syntax:-

```
@ interface Annotation_name
{
 data-type member name1() [default value]
 data-type member name2() [default value].
 -- -- -- --
}
```

Utilization Syntax:-

```
@ Annotation_Name(member_Name1=value1, member_Name2=
 value2,)
```

E

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

→ On the basis of Annotations members there are three types of annotations.

(1) 72

### ① Marker Annotations:-

These are annotations declared without having members

Ex:- Override.

### ② Single Valued Annotations:-

These annotations are declared with exactly one member.

Ex:- @ SuppressWarnings (value = "unchecked")



### ③ Multi Valued Annotations:-

These Annotations are declared with more than one member

Ex:-

@ Table (name = "emp1", immutable = true)

→ In java applications, all the annotations are interfaces, all the annotation interfaces are having a default and common super interface that is "java.lang.annotation.Annotation".

↳ interface name.

→ In java applications there are two types of annotations.

① Standard Annotations

② Custom Annotations.

P.T.O

## ① Standard Annotations:-

These annotations are defined by JAVA programming language and provided along with JAVA predefined library.

There are two types of Standard Annotations.

### ① General purpose Annotations:-

These annotations are very much common and very much frequent in java applications. JAVA has provided all these annotations in the form of "java.lang.package".

Ex:- @Override

@Deprecated

@SuppressWarnings

@FunctionalInterface [Java 8 feature]

L

### ② Meta Annotations:-

These annotations are used to declare other annotations. JAVA has provided all these annotations in a separate package that is "java.lang.annotation" package.

Ex:-

@Inherited

@Documented

@Target

@Retention

≡

### ① @Override:-

In case of method Overriding, we must provide same method prototype to both super class method and sub class method.

In case of method Overriding, if we have any mistakes in sub class method name then compiler will not raise any error even it is failure case of method Overriding. Simply JVM will provide output from super class method.

In the above context, to get an error message describing failure case of method Overriding we have to use @Override annotation.

Ex:-

Class DB-Driver

{

    public void getDriver()

{

    System.out.println("Type-1 Driver");

}

}

Class New-DB-Driver extends DB-Driver

{

    @Override

    public void getDriver()

{

    System.out.println("Type-4 Driver");

}

}

Class Test {

    public static void main(String[] args) {

        DB-Driver driver = new New-DB-Driver();

        driver.getDriver();

}

}

}

→ If we compile the above application, then compiler will provide following error message.

Test.java:10: error: Method does not override or implement  
a method from a supertype.  
    ^  
    @Override  
    ^  
    error.

If we compile the above program then compiler will perform the following actions.

- ① Compiler will recognize @Override annotation.
- ② Compiler will take the respective sub class method name and compare with all the super class methods.
- ③ If no super class method name is matched with sub class method name then compiler will display an error message on command prompt.
- ④ If any super class method name is matched with the respective sub class method name then compiler will not rise any error.

≡

## ② @Deprecated:-

In java, if any method was introduced in JDK 1.0 version and it is having alternative methods in the next versions of java then this method is called as deprecated method and if we access deprecated method in

java applications user compiler will provide depreciation message on command prompt.

java has provided depreciation service for only predefined library by default, JAVA has not provide depreciation service directly to user defined library.

In java applications, if we want to make any user defined method as deprecated method and if we want to display depreciation message on command prompt. When we access deprecated method then we have to use

Ex:-

Class Employee

{ @Deprecated

public void getSalary (int basic, float hK)

{ S.O.P() ("Salary is calculated on the basis of Basic and hK"); }

public void getSalary (int basic, float hK, float Pf, int ta) { }

// ~~Alternative~~ Alternative methods for the above method

S.O.P() ("Salary is Calculated On the Basis of Basic, hK")

Class Test {

psvm (String [] args)

{

Employee emp = new Employee();

emp.getSalary (20000, 25.0f);

}

→ If we compile the above application, Compiler will provide the following deprecation message on command prompt.

```
javac Test.java
```

→ NOTE: \* Test.java uses or Overrides a deprecated API.

NOTE: Recompile with -Xlint:deprecation for details.

≡

### ③ @SuppressWarnings(-)

In java applications, when we perform unsafe or unchecked operations, then Compiler will raise warning messages.

In this context, if we want to hide the generated warning messages then we have to use `@SuppressWarnings` annotation.

Syntax:-

```
@SuppressWarnings("unchecked")
```

Ex:-

In java applications, if we declare any Collection Object without specifying element type or parameter type or generic types then this operation is unsafe operation, where Compiler will raise a warning message.

In this context, to remove warning messages generated by the Compiler then we have to use `@SuppressWarnings` annotation.

≡

Ex:-

```

import java.util.*;
class Employee {
 @SuppressWarnings("unchecked")
 public void getEmps() {
 ArrayList al = new ArrayList();
 al.add("AAA");
 al.add("BBB");
 al.add("CCC");
 al.add("DDD");
 System.out.println(al);
 }
}

class Test {
 public static void main(String[] args) {
 Employee emp = new Employee();
 emp.getEmps();
 }
}

```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

### ④ @FunctionalInterface:-

In java apps, if we declare any interface with exactly one abstract method as a member then that interface is called as functional interface.

Ex:- java.lang.Runnable interface, this interface includes only run() method.

In java applications, if we want to make any interface as functional interface then we have to use `@FunctionalInterface` annotation.

Note:- This feature provided in Java 8.

FR:-

`@FunctionalInterface`

interface I {

void m1();

// void m2(); → error

}

Class A implements I

{

public void m1()

{  
System.out.println("M1-A");

}

Class Test{

public static void main(String[] args){

I i = new A();

i.m1();

}

continued...

Continue..

## Annotations

17-10-2015

(176)

### ① @Inherited :-

In java applications, by default, annotations are not inheritable from Super Class to sub classes. If we provide annotation to the super<sup>Class</sup>, then that annotation is applicable for Only Super Class, not for sub class.

Ex:- @persistable

```
class Employee{
}
class Manager Extends Employee{
}
```

In the above example, Only Employee objects are persistable. Manager Objects are not persistable.

In java applications, if we want to make any annotation as inheritable annotation then we have to declare that annotation with @Inherited meta annotation.

Ex:-

```
@Inherited
@Interface Persistable.
```

```
{
}
@Persistable
```

```
class Employee{
}
```

```
class Manager Extends Employee{
}
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

In the above example, both Employee and Manager Objects are persistable.

## ② @Documented:-

In java applications, if we prepare API documentation for any class by using "javadoc" tool then all the details of the respective class like access modifiers, super class details, implemented interfaces details, variables details, constructor details, method details, ... are included in the generated HTML documentation.

If we declare any annotation for the respective class, then that annotation is not listed in the generated documentation, because all the annotations are not documentable by default.

Ex:-

```
@Persistable
public class Employee {
}
```

D:\apps> javadoc Employee.java

Here @Persistable annotation is not included in the generated HTML documentation.

In java applications if we want to make any annotation as documentable annotation then we have to declare that annotation with @Documented meta annotation.

Ex:-

```
@Documented
@interface Persistable {
}

@Persistable
```

```
public class Employee {
```

(177)

```
D:\apps> javadoc Employee.java
```

Here, html document is able to include @persistable annotation, also along with the metadata of Employee class.

### ③ @Target:-

In general, in java applications annotations are applicable for all the programming elements like Variables, Methods, Constructors, Classes, interfaces, local variables, ...

In java applications, if we want to define elements range to which we want to apply annotations then we have to use @Target annotation.

#### Syntax:-

```
@Target(-----Elements List-----)
```

Here Elements List is provided in the form of the constants like TYPE [ Classes, abstract Classes, interfaces ], FIELD, METHOD, ..... from ElementType enum.

#### Ex:-

```
@Target(ElementType.TYPE, ElementType.METHOD)
```

```
@interface Persistable
```

```
{}
```

```
@Persistable --> Valid
```

```
Class Employee
```

```
{ ----- }
```

```
@Persistable --> Invalid
```

```
private Address addr;
```

② Persistable → Invalid

Employee()

{

}

③ Persistable → Valid

public Address getAccountDetails()

{

}

}{

#### ④ @Retention:-

In java applications, annotations are available upto source code, upto .class files and upto RUNTIME.

If we want to define life time for the annotations then we have to use @Retention annotation.

Syntax:-

@Retention(----- lifetime -----)

where life time is provided in the form of the constants like SOURCE, CLASS, RUNTIME from RetentionPolicy enum.

@Retention(RetentionPolicy.RUNTIME)

④ interface Persistable

{

}

④ Persistable

class Employee

{

}

## ② Custom Annotations:-

These annotations are defined by the developers as per the application requirement.

To implement User defined annotations we have to use the following steps.

### ① Declare user defined annotation by using metadata annotations:

```
import java.lang.annotation.*;
@Documented
@Inherited
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@interface MyAnnotation {
}
```

```
String member1() default "value1";
String member2() default "value2";
```

}

### ② Use Custom Annotation in java application:-

```
@MyAnnotation(member1 = "AAA", member2 = "BBB")
public class MyClass {
}
```

### ③ Create Main Application and access data from Annotation:-

If we apply annotation for classes, methods, variables, . . . they get `java.lang.Class` object and get Annotation object by using the `getAnnotation` method from `Class, method, field, . . .`

```
public Annotation getAnnotation(Class Ann-Type)
```

)

~~Ex:-~~ import java.lang.annotation.\*;  
class Test {  
 public void main(String[] args) throws Exception {  
 Class c = Class.forName("MyClass");  
 Annotation ann = c.getAnnotation(MyAnnotation.class);  
 MyAnnotation ma = (MyAnnotation) ann;  
 System.out.println(ma.member1());  
 System.out.println(ma.member2());  
 }  
~~Ex:-~~ Branch.java:-  
import java.lang.annotation.\*;

- ② Documented
- ② Inherited
- ② Target(ElementType.TYPE)
- ② Retention(RetentionPolicy.RUNTIME)
- ② interfaces Branch {

String bid() default "B-111";  
String bname() default "Ameerpet";  
String mobile() default "91-94471200315";  
}

Account.java:-

② Branch(bid = "B-222", bname = "S R Nagar")  
public class Account {

String accno;  
String accname;  
String accType;

```

Account(String accNo, String accName, String accType)
{
 this.accNo = accNo;
 this.accName = accName;
 this.accType = accType;
}
public void getAccountDetails()
{
 System.out.println("Account Details");
 System.out.println("-----");
 System.out.println("Account Number :" + accNo);
 System.out.println("Account Name :" + accName);
 System.out.println("Account Type :" + accType);

 System.out.println("Account Type :" + accType);
}
}

```

### Test.java:-

```

import java.lang.annotation.*;
Class Test{
 public void main(String[] args) throws Exception{
 Account acc = new Account("abc123", "Durga", "Savings");
 acc.getAccountDetails();
 Class c = Class.forName("Account");
 Annotation ann = c.getAnnotation(Branch.class);
 Branch b = (Branch)ann;
 System.out.println("Branch Details");
 System.out.println("-----");
 }
}

```

S.o.pin("Branch Id : "+b.bid);  
S.o.pin("Branch Name : "+b.bname);  
S.o.pin("Mobile : "+b.mobile());  
} }  
E

## String Manipulations:-

In case of C and C++ applications, to perform String Operations like Concatenation, Identifying length, Converting data from higher case to lower case, ... C and C++ languages are providing some predefined library.

Similarly, to perform String Operations in java applications java has provided predefined library in the form of the following predefined classes.

- java.lang.String
- java.lang.StringBuffer
- java.lang.StringBuilder
- java.util.StringTokenizer

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Q. What is the diff. between String and String Buffer?

- String class Objects are immutable Objects, these Objects are not allowing modifications on their Content directly, these Objects are allowing data to perform operations but the resultant data will not be stored in the Original Objects, where the resultant data will be stored by creating new String Object.
- String class Objects are fixed length Objects.
- StringBuffer class Objects are mutable Objects, these Objects are able to allow modifications on their Content directly.
- StringBuffer class Objects are variable length Objects.

Q. What are the diff. between StringBuffer and String Builder?

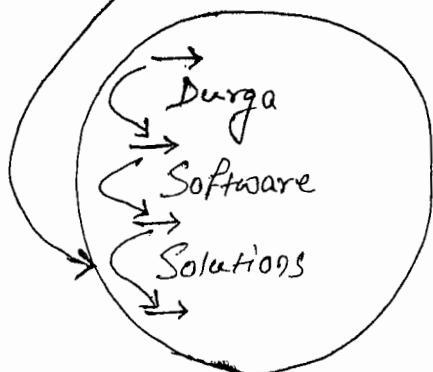
- ① StringBuffer is Synchronized Component.

StringBuilder is non Synchronized Component.

- ② All the methods in StringBuffer are synchronized methods.  
No method is synchronized method in StringBuilder methods.
- ③ StringBuffer is able to allow only one thread at a time to access the data.  
StringBuilder is able to allow more than one thread at a time.
- ④ StringBuffer will give guarantee for data consistency.  
StringBuilder will not give guarantee for data consistency.
- ⑤ StringBuffer is threadsafe.  
StringBuilder is not threadsafe.
- ⑥ StringBuffer is following sequential execution.  
StringBuilder is following parallel execution.
- ⑦ StringBuffer will take more execution time.  
StringBuilder will take less execution time.
- ⑧ StringBuffer will reduce the application performance.  
StringBuilder will improve application performance.
- ⑨ StringBuffer was introduced in jdk1.0 version.  
StringBuilder was introduced in jdk5.0 version.

Q. What is StringTokenizer and how it is possible to perform StringTokenizer in java applications?  
→ The process of generating no. of tokens from a String is called as String Tokenization.  
In general, we will use StringTokenizer to prepare Lexical analyzer in Compiler Construction.

```
String str = "Durga Software Solutions";
 StringTokenizer st = new StringTokenizer(str);
```



```

 boolean b = st.hasMoreTokens();
 if (b == true) {
 System.out.println(st.nextToken());
 }
 b = st.hasMoreTokens();
 if (b == true) {
 System.out.println(st.nextToken());
 }
 b = st.hasMoreTokens();
 if (b == true) {
 System.out.println(st.nextToken());
 }
}

while (st.hasMoreTokens())
{
 System.out.println(st.nextToken());
}
```

→ We will use String Tokenization while converting a number from words representations to numeric representation in bank applications.

→ To perform String Tokenization, JAVA has provided a separate predefined class in the form of `java.util.StringTokenizer`.

→ To Create `StringTokenizer` class Object we have to use the following Constructors.

```
public StringTokenizer(String data)
```

→ It will perform string tokenization on the basis of 'Space delimiter'.

```
public StringTokenizer(String data, String regExp)
```

It will perform String Tokenization on the basis of the specified Reg-Exp delimiter.

Ex:-

```
 StringTokenizer st = new StringTokenizer ("Durga Software Solutions",
```

When Jvm encounters the above instruction, Jvm will take the specified String, Jvm will generate no. of tokens on the basis of 'Space' delimiter and Jvm will store all the tokens in StringTokenizer class object.

Note:- When StringTokenizer class object is created with few no. of tokens then automatically a cursor is created before the first token.

If we want to get no. of tokens which are available in StringTokenizer Object then we have to use the following ~~St~~ method.

```
 public int CountTokens()
```

If we want to retrieve tokens from StringTokenizer class Object then we have to use the following steps.

① Check whether more tokens are available or not from current cursor position.

To perform this action we have to use the following method.

```
 public boolean hasMoreTokens()
```

- It will return the value if atleast next token is available.
- It will return false value if no more tokens are available.

13/10/2015

(182)

- ② If atleast next token is available then read next token and move cursor to next position.  
To perform this action we have to use the following method.

public String nextToken()

Repeat the above two steps upto all the tokens in StringTokenizer Object.

E:

```
import java.util.*;
class Test{
 public void main(String[] args){
 String str = "Durga Software Solutions";
 StringTokenizer st = new StringTokenizer(str);
 int count = st.countTokens();
 System.out.println("No of Tokens :" + count);
 while(st.hasMoreTokens())
 System.out.println(st.nextToken());
 }
}
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## String Class Library:-

### Constructors:-

① public String()

→ This Constructor can be used to create an empty String Object.

Ex:- String str=new String();

② public String(String data)

→ This Constructor can be used to create String class object with the Specified String content.

Ex:-

String str=new String("Durga Software Solutions");  
System.out.println(str);

O/p:- Durga Software Solutions.

When we pass String class Object reference variable as a parameter to System.out.println() method then JVM will access String class toString() method internally, it was implemented in such way that to return a String contains the content of the String Object.

Q: What is the difference between the following two statements?

① String str="abc";

② String str=new String("abc");

=  
P.T.O.

(183)

Ans:-

In java applications, the above two statements can be used to create String Objects.

To Create String class Object, if we use first statement, then String Object will be created in "String Constant pool Area" which is available in Method Area. If any object is created in "String Constant pool Area" then that Objects are not eligible for "Garbage Collection", these Objects are destroyed automatically by JVM when our java program is terminated completely.

To Create String class object, if we use second statement, then one String object is created in heap memory, bcz of "new" keyword and another String object is created in "String Constant Pool Area" with the same data. If any object is created in "String Constant pool Area" then that Object is not eligible for "Garbage Collection", but, if any object is created in heap memory then that Object is eligible for "Garbage Collection".

### (3) public String (byte[]b)

→ This constructor can be used to create String class object with the String equivalent of the Specified byte[] as Content.

Ex:-

```
byte[] b = {65, 66, 67, 68, 69, 70};
String str=new String(b);
System.out.println(str);
```

O/P:- ~~A B C D E F~~

2

- ④ public String (byte[] b, int start\_index, int no\_of\_elements)
- This constructor can be used to create String class object with the string equivalent of the specified byte[] started from the specified start\_index and upto the specified no. of elements.

Ex:-

```
byte[] b = {65, 66, 67, 68, 69, 70};
```

```
String str = new String(b, 3, 2);
```

```
S.o.pn(str);
```

Op:- DE.

- ⑤ public String (char[] ch)

→ This constructor can be used to create String class object with the string equivalent of the specified char[].

Ex:-

```
char[] ch = {'A', 'B', 'C', 'D', 'E', 'F'};
```

```
String str = new String(ch);
```

```
S.o.pn(str);
```

Op:- ABCDEF

- ⑥ public String (char[] ch, int start\_Index, int no\_of\_Elements)

→ This constructor can be used to create String class object with the string equivalent of the specified char[] start from the specified start\_index and upto the specified no. of elements.

Ex:-

```
char[] ch = {'A', 'B', 'C', 'D', 'E', 'F'};
```

```
String str = new String(ch, 3, 2);
```

```
S.o.pn(str);
```

Op:- DE

## Methods:-

(189)

① public int length()

→ It will return size of the String that is no. of characters available in String including Spaces.

Ex:-

String str=new String("Durga Software Solutions");

S.o.p1n(str.length());

O/P:- 24

② public String concat(String str)

→ This method can be used to add the specified string to the String object content in immutable manner.

Ex:-

String str1=new String("Durga ");

String str2=~~str1~~.concat("Software ");

String str3=str2.concat(" Solutions");

S.o.p1n(str1);

S.o.p1n(str2);

S.o.p1n(str3);

O/P:- Durga

Durga Software

Durga Software Solutions.

Ex:-

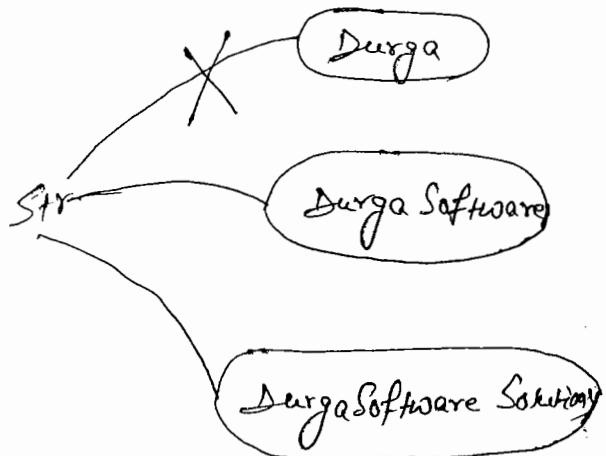
String str="Durga ".concat("Software ").concat(" Solutions")  
S.o.p1n(str);

O/P:- Durga Software Solutions

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

E  
P.T.O.

String str = "Durga ".concat("Software ").concat("Solutions");



O/P:- Durga Software Solutions.

Continued...

(185)

## String Manipulations

14/10/2015

Continued...

Public

### ③ boolean equals(Object obj)

→ This method can be used to check whether two String Object Contents are same or not. If they are same then equals() method will return true value. If they are not same then equals() method will return false.

Ex:-

3

```
Ex- class Test {
 public static void main(String[] args) {
 String str1 = new String("abc");
 String str2 = new String("def");
 String str3 = new String("abc");
 System.out.println(str1.equals(str2));
 System.out.println(str2.equals(str3));
 System.out.println(str3.equals(str1));
 }
}
```

O/P:- false

false

true.

Q: What is the difference between == Operator and equals() method?

→ In java, '==' is basically a comparison operator, it will check whether the provided two Operands values are same or not, where the Operands may be primitive variables or object reference Variables.

Initially, equals() method provided in java.lang.Object class, it was implemented in such a way that to provide comparison between two object reference values, it will not provide comparison between two Objects contents. As per the String class requirement, String class is not using Object class equals() methods, String class has Overrides

Object class equals() method is such a way that to perform the comparison between two String object content instead of performing comparison between two String Object reference values.

Ex:- Class A {

```

 CLASS Test {
 PSVM (String [7 args]) {
 int i=10;
 int j=20;
 A Q1 = new A();
 A Q2 = new A();
 String str1 = new String ("abc");
 String stro = new String ("def");
 String str2 = new String ("abc");
 S.O.Pln (i==j);
 S.O.Pln (Q1==Q2);
 S.O.Pln (str1==str2);
 S.O.Pln ();
 S.O.Pln (Q1.equals (Q2));
 S.O.Pln (str1.equals (str2));
 }
 }

```

Opp:- false

false

false

false

true.

#### ④ Public boolean equalsIgnoreCase(String str)

→ In java applications, equals() method will perform case sensitive comparision between two String Object Contents, equalsIgnoreCase() method will perform case Insensitive Comparision between two String <sup>Object</sup> Contents.

Ex:-

```
class Test {
 public static void main(String[] args) {
 String str1 = new String("abc");
 String str2 = new String("ABC");
 System.out.println(str1.equals(str2));
 System.out.println(str1.equalsIgnoreCase(str2));
 }
}
```

#### ⑤ Public int compareTo(String str)

→ This method will check whether two String Object Contents are available in dictionary order or not.

str1.compareTo(str2)

① If str2 come first when compared with str1 in dictionary order then compareTo() method will return +ve value.

② If str2 come next when compared with str1 in dictionary order then compareTo() method will return -ve value.

③ If both str1 and str2 are same and if they are available at the same position in dictionary order then compareTo() method will return 0 value.

```
Ex:- Class Test {
 public static void main(String[] args) {
 String str1 = new String("abc");
 String str2 = new String("def");
 String str3 = new String("abc");
 System.out.println(str1.compareTo(str2));
 System.out.println(str2.compareTo(str3));
 System.out.println(str3.compareTo(str1));
 }
}
```

O/P:-  
-3  
0

#### ⑥ public boolean startsWith(String str)

→ It will check whether a String start with the specified String or not.

#### ⑦ public boolean endsWith(String str)

→ It will check whether a String ends with the specified String or not

#### ⑧ public boolean contains(String str).

→ It will check whether a String contains the specified String or not.

```
Ex:- Class Test {
 public static void main(String[] args) {
 String str = new String("Durga Software Solutions");
 System.out.println(str.startsWith("Durga"));
 System.out.println(str.endsWith("Solutions"));
 System.out.println(str.contains("Software"));
 }
}
```

O/P:-  
true  
true  
true

⑨ public int indexOf(String str)

→ This method will return an index value where the first occurrence of the specified string is available.

⑩ public int lastIndexOf(String str)

→ This method will return an index value where the last occurrence of the specified string is available.

Ex:-

```
class Test{
 public static void main(String[] args){
 String str=new String("Durga Software Solutions");
 System.out.println(str.indexOf("so"));
 System.out.println(str.lastIndexOf("so"));
 }
}
```

O/P:- 6  
15

⑪ public char charAt(int index)

→ This method will return a particular character available at the specified index value.

Ex:-

```
String str=new String("Durga Software Solutions");
System.out.println(str.charAt(6));
```

O/P:- S

⑫ public String replace(char oldChar, char newChar).

→ This method will replace ~~with~~ the specified old character with the specified new character in a string.

Ex:- String str=new String("Durga Software Solutions");

```
System.out.println(str);
System.out.println(str.replace('S','s'));
```

O/P:- Durga Software Solutions  
Durga software Solution

16/10/2015 - (188)

(13) `public String substring(int start_index)`

→ This method will generate a substring from a string starts from the specified start index.

(14) `public String substring(int start_index, int end_index)`

→ This method will generate a substring from a string starts from the specified start index and ends upto the specified end index.

Ex:-

Class Test {

    public void main(String[] args) {

        String str = new String("Durga Software Solutions");

        System.out.println(str.substring(6));

        System.out.println(str.substring(6, 14));

}  
}

(15) `public byte[] getBytes()`

→ This method convert a String into ASCII values in the form of byte[].

Ex:- Class Test {

    public void main(String[] args) {

        String str = new String("Durga Software Solutions");

        byte[] b = str.getBytes();

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

            System.out.print(b[i] + " --- " + (char)b[i]);

}

}  
}

### ⑯ public char[] toCharArray()

16/10/2015

→ This method will Convert a String into Char[].

Ex: Class Test{

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

```
String str = new String ("Durga Software Solutions");
```

```
char[] ch = str.toCharArray();
```

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

```
{ System.out.print(ch[i] + " "); }
```

### ⑰ public String trim()

→ This method will remove before Spaces and after Spaces of a particular String.

{

```
String str = new String (" Durga Software Solutions ");
```

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

```
System.out.println(str.trim());
```

### ⑱ public String toUpperCase()

→ It will Convert all the characters of a String into Upper Case letters.

### ⑲ public String toLowerCase()

→ It will Convert all the characters of a String into Lower Case letters.

Ex:-

```

String str = new String("Durga Software Solutions");
S.o.pIn(str.toUpperCase());
S.o.pIn(str.toLowerCase());

```

### ⑩ public String[] split(String reg-Expr)

→ This method will divide a String into no. of tokens on the basis of the specified Reg-Expr.

Ex:-

```

class Test {
 public static void main(String[] args) {
 String str = new String("Durga
String[] s = str.split(" ");
for (int i = 0; i < s.length; i++) {
 S.o.pIn(s[i]);
 }
}

```

### \* StringBuffer

→ Constructors:-

#### 1) public StringBuffer()

→ This Constructors will Create an empty StringBuffer Object with the default initial capacity 16 elements.

```

StringBuffer sb = new StringBuffer();
S.o.pIn(sb.capacity());

```

O/P:- 16

② public StringBuffer(int capacity)

→ It will create an empty StringBuffer Object with the specified capacity value.

StringBuffer sb = new StringBuffer(10);

S.0.P19 (sb.capacity());

OP:- 10

③ public StringBuffer(String data)

→ This Constructor will create StringBuffer object with the specified String content.

Ea:- class Test {

  public static void main(String[] args) {

    StringBuffer sb = new StringBuffer("abc");

    S.0.P19 (sb);

    S.0.P19 (sb.capacity());

  }

OP:- abc  
19

→ When we pass StringBuffer Object reference variable as parameter to System.out.println() method. then internally jvm will access StringBuffer class toString() instead of executing object class toString() method.

→ In StringBuffer class, toString() method was implemented in such a way that to return the content of StringBuffer Object in the form of String.

→ if we provide String content to StringBuffer Object then the capacity of the StringBuffer obj will be increased with the full ratio.

NewCapacity = InitialCapacity + length of String.

Methods:-

① public StringBuffer append(String str)

→ This method will append the specified String to the StringBuffer object content & it's mutability nature.

Ex:-

Class Test {

PSVM (String[] args) {

String str1 = new String ("Durga");

String str2 = str1.concat ("Software");

String str3 = str2.concat ("Solutions");

S.O.P (str1);

S.O.P (str2);

S.O.P (str3);

S.O.P ();

StringBuffer sb1 = new StringBuffer ("Durga");

sb2 = sb1.append ("Software");

sb3 = sb2.append ("Solutions");

S.O.P (sb1);

S.O.P (sb2);

S.O.P (sb3);

}

2

② public void ensureCapacity (int capacity)

→ This method will set the specified capacity value to StringBuffer object explicitly.

If the capacity value is less than 16 then StringBuffer object will take 16 as capacity value.

If the capacity value is greater than 16 and less than 34 then the capacity value is 34.

If the capacity value is 34 then the specified value is the capacity value for StringBuffer.

Ex:-

```
Class Test {
 Psvm (String[] args) {
 StringBuffer sb = new StringBuffer();
 sb.ensureCapacity(10);
 System.out.println(sb.capacity());
 }
}
```

③ public StringBuffer reverse()

→ This method will reverse the content of StringBuffer class object.

Ex:-

```
StringBuffer sb = new StringBuffer("Durga Software Solutions");
```

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

```
System.out.println(sb.reverse());
```

O/P:- Durga Software Solutions

solutions erawtfos agrud

④ public StringBuffer insert (int index, String data)

→ It will insert the specified string at the specified index value in StringBuffer Object.

Ex:-

```
StringBuffer sb = new StringBuffer("Durga Solutions");
```

```
System.out.println(sb.insert(6, "Software"));
```

O/P:- Durga Software Solutions

(191)

- ⑤ public StringBuffer delete( int start-index, int end-index )  
→ It will delete a String from StringBuffer Object available  
in the specified range.

StringBuffer sb = new StringBuffer("Durga Software Solutions");  
System.out.println(sb.delete(6, 14));

O/P :- Durga Solutions.

~~String manipulation finished.~~

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

26/10/2015

## Packages:-

package is a collection of related classes and interfaces as a single unit.

package is a folder contains .class files representing related classes and interfaces.

In java applications, packages will provide the following advantages.

### ① Modularity:-

In java applications, module is a folder which contains .class files representing related classes and interfaces, so that, to declare <sup>modules</sup> ~~classes~~ in java apps we have to use package.

### ② Abstraction:-

In java apps, if we declare classes and interfaces in a package then that classes and interfaces are not available to outside, so, that packages are able to improve abstraction.

### ③ Security:-

In java application, packages are able to provide encapsulation and abstraction, so, that package are able to provide security.

### ④ Reusability:-

In java applications, declare a package one time and we are able to use that package any no. of times, so that, packages are able to improve reusability.

### ⑤ Sharability:-

In java applications, declare package one time and we are able to share that package to multiple java programs.

at a time, so that, packages are able to improve Sharability.

There are two types of packages in java applications.

- ① Pre-Defined packages
- ② User Defined packages.

### ① Pre-Defined packages:-

These packages are defined by JAVA programming language and provided along with java software.

Ex:- java.lang:-

This package is a default package in java files, no need to import this package in java applications. This package contains all the basic classes and interfaces which are used frequently in all the java programs.

This includes the classes and interfaces like String, StringBuilder, Exception and its sub classes, Thread, Runnable, ...

Ex:- ② java.io:-

This package has provided predefined classes and interfaces to perform IO Operations in java applications.

This package has provided the classes and Interfaces like Reader, Writer, InputStream, OutputStream, FileInputStream, FileOutputStream, DataInputStream, DataOutputStream, ...

Ex:- ③ java.util:-

This package contains predefined classes and interfaces to represent all the Collection Objects or data structures. To represent data structures, java.util package has

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

provided the classes and interfaces like,

List, Set, Queue, Map, SortedSet, NavigableSet, SortedMap, NavigableMap, ArrayList, Vector, LinkedList, Stack, ...

Ex(4) :- java.net :-

This package has provided predefined classes and interfaces to provide socket programming in order to design distributed applications.

This package includes the classes and interfaces like Socket, ServerSocket, URL,URLConnection, InetAddress, InetSocketAddress, ...

Ex(5) :- java.awt :-

This package contains all the predefined classes and interfaces to prepare GUI applications. This package contains the classes and interfaces like TextField, Button, CheckBox, List, Choice, ...

Ex(6) :- javax.swing :-

In java applications, we are able to use AWT, SWING to prepare GUI applications, AWT provided components are heavy weight, platform dependent, peer components. To overcome these problems, JAVA has redesigned all the GUI components which are provided by AWT in the form of javax.swing package.

javax.swing package contains the classes and interfaces like JLabel, JButton, JTextField, JPasswordField, JCheckBox, JRadio, ...

Ex(7) :- java.sql :-

This package contains the predefined library to design JDBC applications.

java.sql package includes the classes and interfaces like

## Connection

Driver

DriverManager

Statement

CallableStatement

ResultSet

-----  
-----

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## ② User Defined Packages:-

These packages are provided by the developers as per their application requirement.

To declare user defined packages in java applications we have to use the following syntax:

package package-name;

If we want to use package declaration statement in java files then we have to use the following two conditions.

- ① Always, package declaration statement must be provided as first statement in java application (files).
- ② Package name must be unique, it must not be sharable and it must not be duplicated.

Q: Is it possible to provide more than one package declaration statement with in a single java file?

→ No, it is not possible to provide more than one package declaration statement, because package declaration statement must be first statement in java files.

≡

To give package names, JAVA has provided a convention like to include our company domain name in reverse.

Ex:-

```
package com.deergasoft.eeeli.transactions;
```

Com.deergasoft → Company domain name in reverse.

eeeli → Client/project name

transaction → Modelle name.



27-10-2015

\* If we declare class and interfaces in package or if we want to use them in present java file then we have to import package to the present java file.

To import a particular package in present java file, we have to use the following syntaxes:-

```
import package-name.*;
```

It able to import all the classes and interfaces of the specified package into the present java file.

Ex:-

```
import java.util.*;
```

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

Syntax:- `import package-name.member-name;`

It able to import only the specified class or interface in the present java file.

Ex:- `import java.util.ArrayList;`

```
import java.io.BufferedReader;
```

Note:- In java files, we are able to provide almost one package declaration statement. But, we are able to provide any no. of import statements.



Dilaps

Employee.java

delete \* . class (194)

C:\XY2

Com

|-- durgsoft

|-- Core

|--- Employee.class

D:\javatam

MainApp.java

MainApp.class

~~E:\~~ D:\apps\Employee.java:

```
package com.durgsoft.core;
public class Employee {
 String eid;
 String ename;
 float esal;
 String eaddr;
 public Employee(String eid, String ename, float esal, String eaddr)
 {
 this.eid = eid;
 this.ename = ename;
 this.esal = esal;
 this.eaddr = eaddr;
 }
 public void getEmpDetails()
 {
 System.out.println("Employee Details");
 System.out.println("-----");
 System.out.println("Employee Id: " + eid);
 System.out.println("Employee Name: " + ename);
```

```

 System.out.println("Employee Salary :" + esal);
 System.out.println("Employee Address :" + eaddr);
 }
}

```

D:\apps> javac -d C:\xyz Employee.java

D:\javatam\MainApp.java.

```

import com.deergasoft.core.*;
public class MainApp {
 public static void main(String[] args) {
 Employee emp = new Employee("E-111", "Durga", 5000, "Hyd");
 emp.getEmpDetails();
 }
}

```

To compile the above java file, we must set "CLASSPATH" environment variable to the location where "com.deergasoft.core" folder structure is existed that is "C:\xyz" location.

D:\javatam> set CLASSPATH=c:\xyz; (from this command we can replace our file in other locations which we want).

D:\javatam> javac MainApp.java.

D:\javatam> java MainApp.

----- Output ----- ≡

Ex:- D:\My application\app.jar  
 (folder name) (filename)

JAR files in java:-

If we want to transfer a set of .class files from one machine to another machine and if we want to upload and download a set of .class files then it is not suggestable to transfer or upload and download individual .class files.

Continued... packaged. 27/10/2015-

(195) 28/10/2015

files, here we have to use JAR file with the .class files, then we can transfer or upload and download JAR file.

To Create JAR file, we have to use the following command.

→ jar -cvf file-name.jar \*.\*

To extract JAR file, we have to use the following command.

→ jar -xvf file-name.jar

C → Create

V → Verbose

f → File

X → Extract

### D:\my applications

Customer.java

Compile the above java file and send the generated .class files to C:\abc location.

### C:\abc

com

|--- dergasoft

|--- Core

|--- Customer.class

prepare jar for the above application and copy to D:\xyz

D:\xyz

(app.jar) → file name.

D:\javatm

MainApp.java

MainApp.class

Set classpath to jar file that is "D:\xyz\app.jar, Compile MainApp.java and execute MainApp class.

## D:\myapplications\Customer.java

```
package com.devgasoft.core;
public class Customer {
 private String cid;
 private String cname;
 private String caddr;
 public Customer(String cid, String cname, String caddr) {
 this.cid = cid;
 this.cname = cname;
 this.caddr = caddr;
 }
 public void getCustomerDetails() {
 System.out.println("Customer Details");
 System.out.println("-----");
 System.out.println("Customer Id :" + cid);
 System.out.println("Customer Name :" + cname);
 System.out.println("Customer Address :" + caddr);
 }
}
```

To compile this program,

D:\myapplications>javac -d C:\abc Customer.java

C:\abc>jar -cvf app.jar \*.\*

Copy the generated jar file to D:\xyz

## D:\javafam\MainApp.java

```
import com.devgasoft.core.*;
public class MainApp {
 public static void main(String[] args) {
 Customer c = new Customer("C-111", "Devgal", "Hyd");
 c.getCustomerDetails();
 }
}
```

(96)

D:\javatam> set Classpath = D:\xyz\app.jar ;

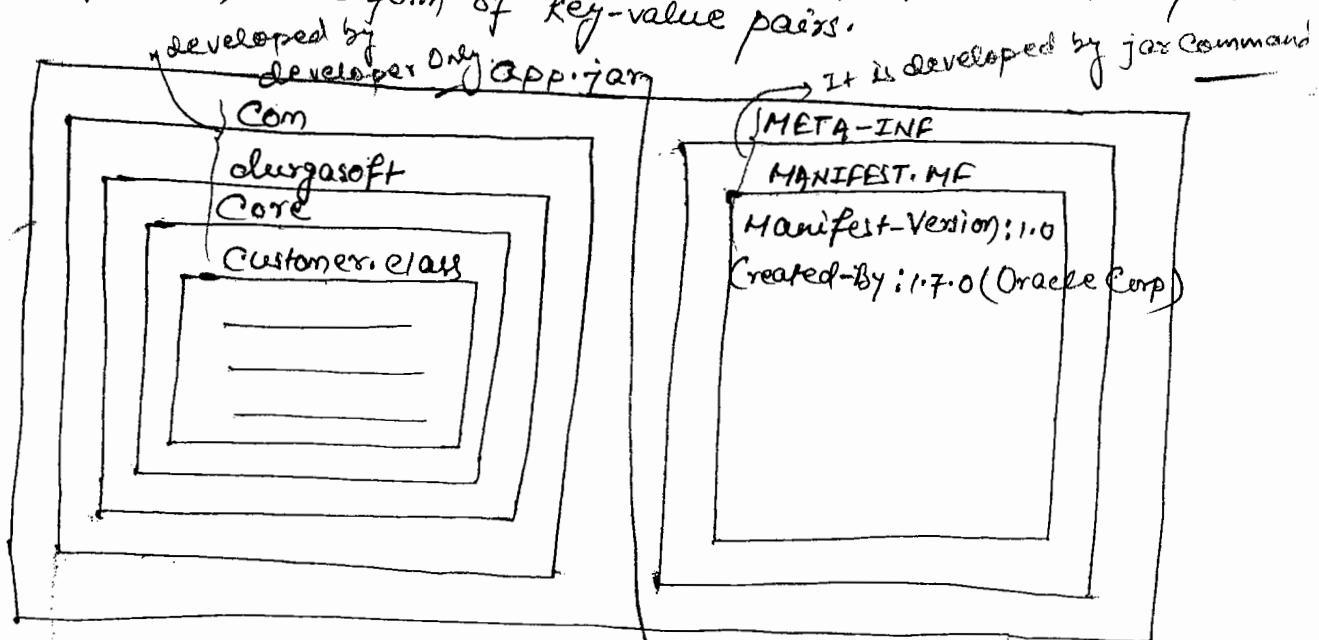
D:\javatam> javac MainApp.java

D:\javatam> java MainApp

----- Customer Details -----  
=====

### MANIFEST File in Jar file:-

MANIFEST.MF is a file created by "jar" command at the time of creating jar file and it is located in META-INF folder, jar file is in the form of key-value pairs.



→ first we have to create the application and save it → after that we can change the location and in the new location we have to extract this file for META-INF.

### Executable JAR file:-

Executable JAR file is a jar file containing main class and main() method.

To prepare executable JAR file in java application we have to use the following steps.

- ① Create Application and compile the required java files.
- ② Declare "Main-Class" attribute in a text file.
- ③ Create jar file with user defined data e) MANIFEST.MF file
- ④ Execute jar file.

Ex:- D:\javatutorial\app\MainApp.java.

```

import javax.swing.*;
class Logoframe extends Frame {
 Logoframe() {
 this.setVisible(true);
 this.setSize(700, 500);
 this.setTitle("Logo Frame");
 this.setBackground(Color.green);
 }
 public void paint(Graphics g) {
 Font f = new Font("arial", Font.BOLD, 30);
 g.setFont(f);
 g.drawString("Durga Software Solutions", 50, 100);
 }
}
class MainAPP {
 public static void main(String[] args) {
 Logoframe lf = new Logoframe();
 }
}

```

Continued..

D:\javafam\app\abc.txt

Main-Class: MainApp

The main extension of this file is to keep Main-Class attribute in MANIFEST.MF file in order to give an information to the JVM about the main class name to execute app.

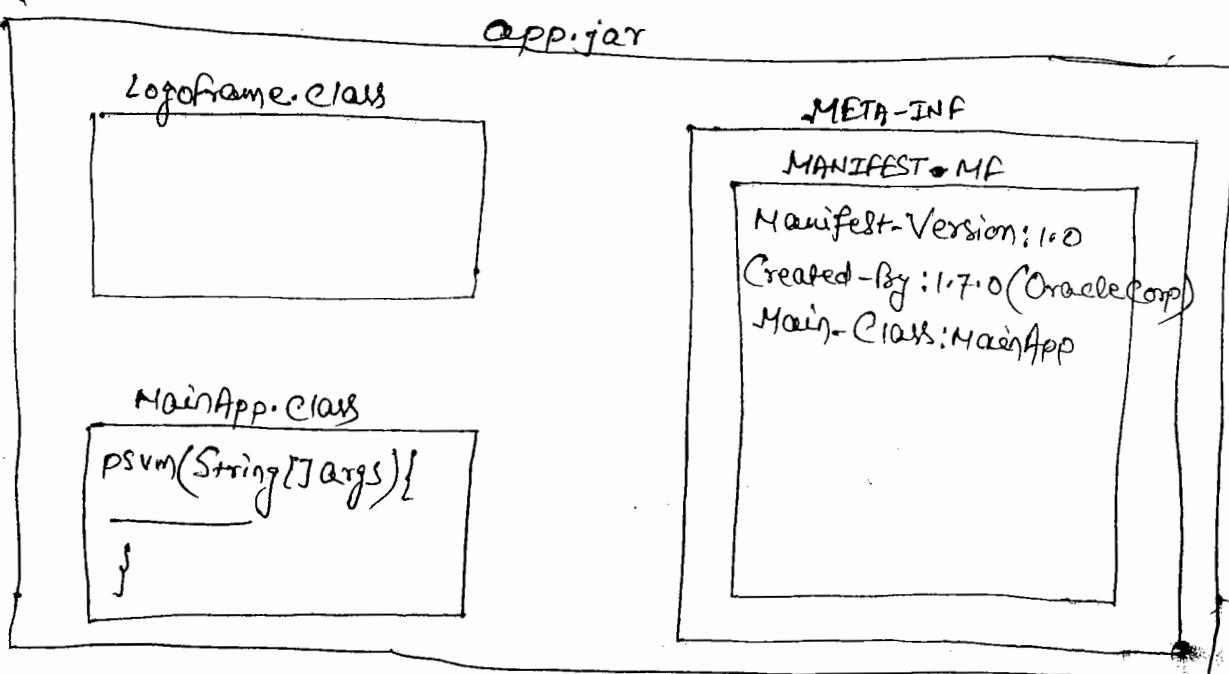
D:\javafam\app>javac MainApp.java

D:\javafam\app>jar -cvfm app.jar abc.txt \*.\*

D:\javafam\app>java -jar app.jar

When we execute jar file with the above command then JVM will perform the following actions.

- ① JVM will recognize -jar option and take app.jar file.
- ② JVM will go to MANIFEST.MF file available under "META-INF" folder inside app.jar file and get "Main-Class" attribute value that is Main class name.
- ③ After getting Main class name from MANIFEST.MF file, JVM will execute Main Class.



29/10/2015

If we want to execute the above application by using batch file then we have to use the following steps.

① Take a Text file with the execution command.

java -jar app.jar

② Save text file with "app.bat".

③ Double click on bat file.

Note:- jar file and bat file must be available at the same loc.

D:\javazam\app\app.bat → java -jar app.jar

On command prompt

→ Automatically executed when we double clicked on bat file.  
("bat" file is displayed on desktop window)

packages finished

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

By using java programming language, we are able to prepare the following two types of applications.

- (1) CUI Applications
- (2) GUI Application.

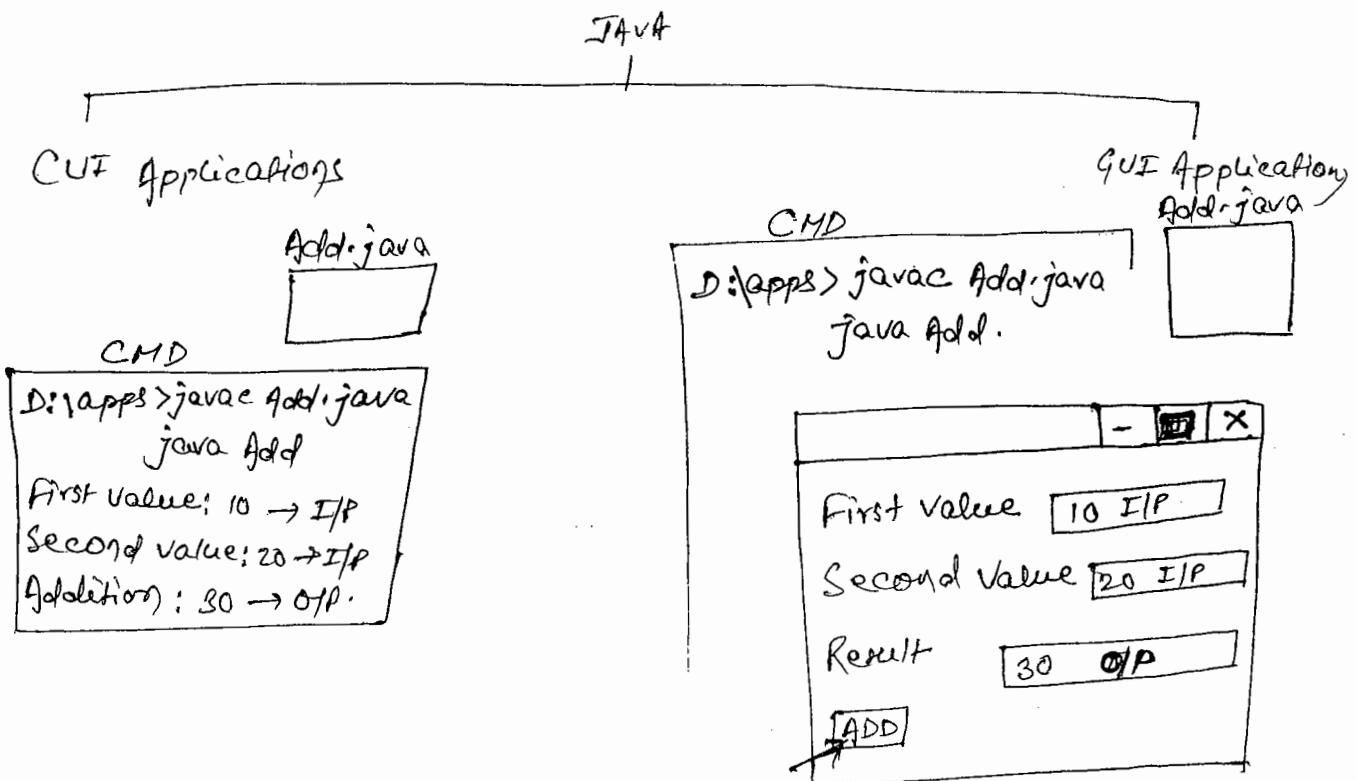
Q. What is the diff. between CUI and GUI applications?

→ CUI Application is a java application, it would be designed in such a way that to take input through command prompt and to provide output through command prompt, where command prompt is acting as an user interface and it is able to support only character Oriented data.  
To prepare CUI Applications we have to use the core libraries like java.io, java.util, java.lang, ... packages.

→ GUI Application is a java application, it would be designed in such a way that to take input through a collection of Graphics Components and to provide output through the same collection of Graphics Components, where the collection of Graphics component is acting as an user interface and it is able to support Graphics data.  
To prepare GUI Applications, JAVA has provided the predefined library in the form of the following three packages.

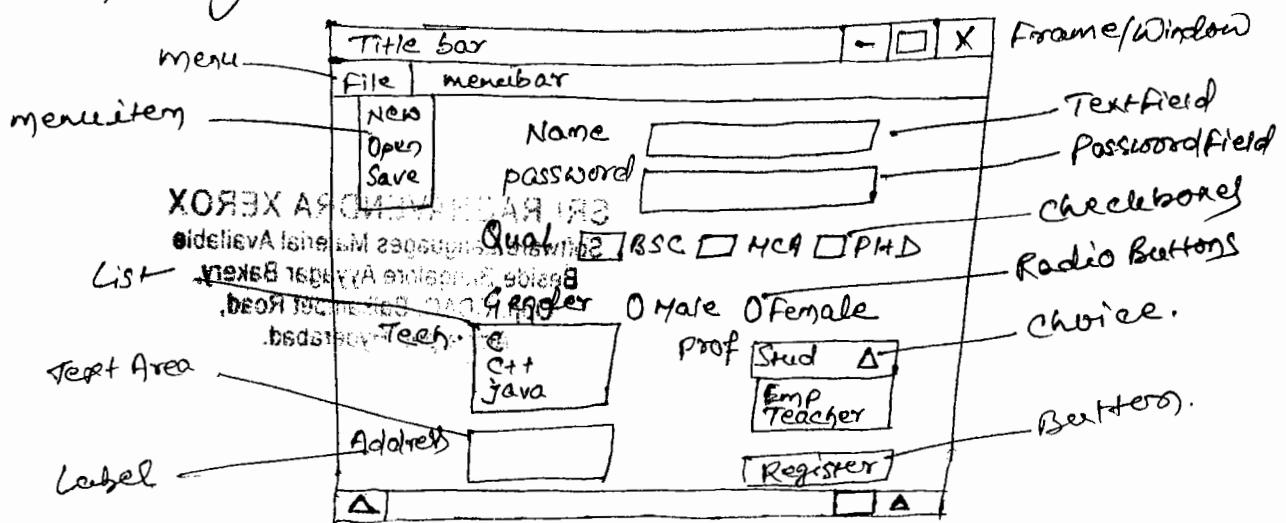
- (1) java.awt
- (2) java.applet
- (3) javax.swing.

Z



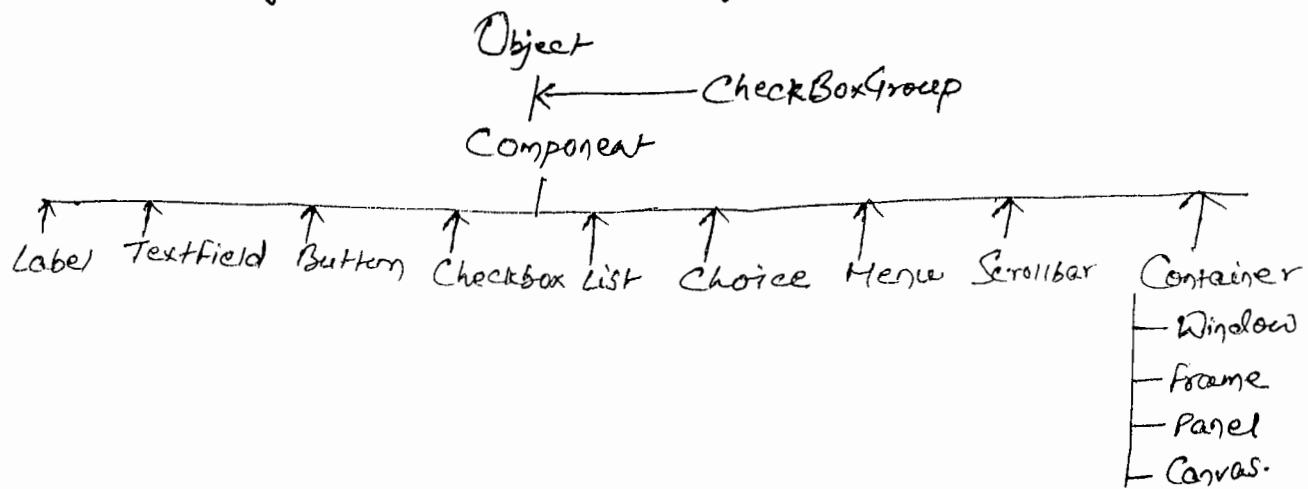
### AWT (Abstract Windowing Toolkit) :-

- AWT is a framework, a technology, a collection of predefined classes and interfaces, which will provide very good environment to prepare GUI Applications or Desktop Applications or Window Based Applications.
- AWT is able to provide predefined library to represent the following GUI components.



(2)

To represent all the above GUI components, AWT has provided the following predefined library as part of `java.awt` package.



### Container :-

14-10-2015

Container is a GUI Component, it contains some other GUI components like labels, Textfields, checkboxes, ...

Ex:- Frame

Window  
panel  
Canvas.

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### Frame:-

To represent frame container, AWT has provided a separate predefined class in the form of `java.awt.Frame`.

To create frame class object we have to use the following constructors.

`public Frame()`

→ It will create a frame without title.

`public Frame(String title)`

→ It will create a frame with the specified title.

Ex:- `Frame f=new Frame();`

Creating Frame class object is not sufficient to get a frame physically, if we want to get frame physically we have to use either of the following methods.

public void show()

→ It is deprecated method, it will show the frame but, it is unable to hide the frame.

public void setVisible(boolean b)

→ If 'b' value is true then frame will be in visible mode. If 'b' value is false then frame will be in invisible mode.

When we create a frame then frame will have 0-width and 0-height size by default, where if we want to set a particular size to the frame we have to use the following method.

public void setSize(int width, int height)

If we want to provide a particular title to the frame explicitly then we have to use the following method.

public void setTitle(String title)

If we want to set a particular background color to the frame we have to use the following method.

public void setBackground(int color)

where color may be Constant Variables like red, blue, green, ... from java.awt.Color class.

≡

(3)

```

Ex:- import java.awt.*;
class FrameEx {
 public static void main(String[] args) {
 frame f=new frame();
 // f.show();
 f.setVisible(true);
 f.setSize(500, 500);
 f.setTitle("Frame Example");
 f.setBackground(Color.green);
 }
}

```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

In GUI applications, the above approach is not suggestible to prepare frames, because we are unable to provide customizations on frame. To provide customizations on frame we have to create user defined frames, for this we have to use the following steps.

- ① Declare an user defined class.
- ② Extend User defined class from java.awt.frame class.
- ③ Provide all the frame properties by taking constructor in user defined frame class.
- ④ In main class, in main() method, create object for user defined frame class.

Ex:-

```

import java.awt.*;
class Myframe extends frame{
 Myframe()
 {
 this.setVisible(true);
 this.setSize(500, 500);
 this.setTitle("Customized frame");
 this.setBackground(Color.red);
 }
}

```

Class Frame

Method actionPerformed(Event e)

PSUM(String args)

MyFrame mf = new Myframe();

2

If we want to display a text message on frame, we have to override frame class paint() method in user defined frame class.

public void paint(Graphics g)

In paint() method,

① Create font class object with three font properties.

Font f = new Font("Arial", Font.BOLD, 30);

Where "Arial" is font type

where Font.BOLD is font style

where 30 is font size.

② Set font class object to Graphics object.

g.setFont(f);

③ Display a text message on frame

g.drawString("Welcome", 100, 100);

where "Welcome" is text message.

where 100, 100 are x-axis and y-axis location.

2 3

P.T.O

(7)

```
import java.awt.*;
class MyFrame extends Frame {
 MyFrame() {
 this.setVisible(true);
 this.setSize(700, 400);
 this.setTitle("Customized frame");
 this.setBackground(Color.green);
 }
 public void paint(Graphics g) {
 Font f = new Font("arial", Font.BOLD + Font.ITALIC, 40);
 g.setFont(f);
 this.setForeground(Color.red);
 g.drawString("DURGA SOFTWARE SOLUTIONS", 100, 100);
 }
}
class FrameEx {
 public static void main(String[] args) {
 MyFrame mf = new MyFrame();
 }
}
```

### Flow of Execution:-

In the above application, when we create Object for user defined frame class, JVM will execute Super class [Frame class] 0-Arg Constructor before executing user defined frame class constructor. Then in frame class constructor, JVM will access repaint() method, where repaint() method will access paint() method which we defined in user defined frame class.



15/10/2015

## Event Handling / Event Delegation Model:-

In GUI Applications, when we click on a button, checkboxes, radio buttons, list box, . . . . . automatically GUI Apps are performing some actions.

In this Context, GUI Components are not performing any action in their own way, GUI applications are able to raise some events when we trigger that GUI Components.

In GUI Applications, we have to use listeners to handle the events which are raised by the GUI Components. To handle the events which are raised by the GUI components by delegating them to listeners we have to use event handling.

To perform event handling in GUI Applications, the completed predefined library is provided by Java in the form of `java.awt.event package`.

To perform Event handling in GUI Applications, we have to use the following elements.

| <u>GUI Component</u> | <u>Event Type</u> | <u>Listener Type</u> | <u>Listener Methods</u>                                      |
|----------------------|-------------------|----------------------|--------------------------------------------------------------|
| Button               | ActionEvent       | ActionListener       | pv actionPerformed(ActionEvent e)                            |
| Menu                 | "                 | "                    | "                                                            |
| TextField            | FocusEvent        | FocusListener        | pv focusGained(FocusEvent fe)<br>pv focusLost(FocusEvent fe) |
| TextArea             | "                 | "                    | "                                                            |
| CheckBox             | ItemEvent         | ItemListener         | pv itemStateChanged(ItemEvent ie)                            |
| RadioButton          | "                 | "                    | "                                                            |
| List                 | "                 | "                    | "                                                            |
| Choice               | "                 | "                    | "                                                            |

Scrollbar Adjustment Adjustment p v adjustmentValueChanged(ActionEvent ae)

Window WindowEvent WindowListener p v windowOpened(WindowEvent we)  
p v windowClosed(WindowEvent we)  
p v windowClosing(" ")  
p v windowIconified(" ")  
p v windowDeiconified(" ")  
p v windowActivated(" ")  
p v windowDeactivated(" ")

Mouse MouseEvent MouseListener p v mouseClicked(MouseEvent me)  
p v mousePressed(MouseEvent me)  
p v mouseReleased(MouseEvent me)  
p v mouseEntered(MouseEvent me)  
p v mouseExited(MouseEvent me)

Keyboard KeyEvent KeyListener p v keyTyped(KeyEvent ke)  
p v keyPressed(KeyEvent ke)  
p v keyReleased(KeyEvent ke)

### Steps To implement Event Handling in GUI Applications:-

① Create GUI Component in a Container Class Constructors:-

Button b=new Button("Login");

② Identify the respective Listener and implement Listener interface in a class [Container class]:-

Class MyFrame extends Frame implements ActionListener  
{

-----  
public void actionPerformed(ActionEvent ae)

16/10/2015



### ③ Attach ActionListener implementation to GUI Component:-

→ To attach listener implementation to GUI component we have to use the following method.

public void addxxxListener(XXXListener l)

where xxxListener may be ActionListener, ItemListener, -----

b. addActionListener(new Myframe());

or

b. addActionListener(this);

Ex:- import java.awt.\*;

import java.awt.event.\*;

Class WindowListenerImpl implements WindowListener {

public void windowOpened(WindowEvent we) {  
System.out.println("WindowOpened");  
}

public void windowClosed(WindowEvent we) {  
System.out.println("WindowClosed");  
}

public void windowClosing(WindowEvent we) {  
System.exit(0);  
}

System.out.println("WindowClosing");  
}

public void windowIconified(WindowEvent we) {  
System.out.println("WindowIconified");  
}

(6)

```

public void windowDeiconified(WindowEvent we)
{
 System.out.println("WindowDeiconified");
}
public void windowActivated(WindowEvent we)
{
 System.out.println("WindowActivated");
}
public void windowDeactivated(WindowEvent we)
{
 System.out.println("WindowDeactivated");
}

class MyFrame extends Frame {
 MyFrame()
 {
 this.setVisible(true);
 this.setSize(500,500);
 this.setTitle("Window Events");
 this.setBackground(Color.red);
 }
}

```

```

WindowListenerImpl wi = new WindowListenerImpl();
this.addWindowListener(wi);

```

```

class WindowEventsEx {
 public static void main(String[] args) {
 MyFrame mf = new MyFrame();
 }
}

```

Ex

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

### WindowEvent:-

In the above GUI application if we want to provide Only window Closing Option to the frame then it is unnecessary to implement all the remaining methods in WindowListenerImpl Class, Only it is required to implement windowClosing() method.

In the above context, to remove unnecessary methods implementations in WindowListenerImpl AWT has provided an Adapter class in the form of java.awt.event.WindowAdapter abstract class.

If we want to use WindowAdapter abstract class in GUI applications then we have to extend WindowAdapter abstract class in an user defined class and we have to Override only the

### Ex:-

```
import java.awt.*;
import java.awt.event.*;
```

```
Class WindowListenerImpl extends WindowAdapter {
 public void windowClosing(WindowEvent we){
 System.exit(0);
 } }
```

```
Class Myframe extends Frame {
```

```
 Myframe(){
 this.setVisible(true);
 this.setSize(500,500);
 this.setTitle("Window Events");
 this.setBackground(Color.red);
 }
```

```
 WindowListenerImpl wl = new WindowListenerImpl();
 this.addWindowListener(wl);
}
```

(7)

```

class WindowEventsEx
{
 public void main(String[] args)
 {
 MyFrame mf = new Myframe();
 }
}

```

In the above GUI applications, to provide windowClosing() method we have taken a separate class like WindowListenerImpl, it is unnecessary in GUI applications, it will increase no. of class files in GUI applications. In this context, to remove unnecessary classes in GUI applications we have to use anonymous inner class of WindowAdapter abstract class as parameter to addWindowListener() method.

By the use of Anonymous Inner Class.

```

this.addWindowListener(new WindowAd-
-apter()
{
 public void windowClosing(WindowEvent we)
 {
 System.exit(0);
 }
});

```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

```
Ex:- import java.awt.*;
import java.awt.event.*;
```

```
Class MouseListenerImpl implements MouseListener {
public void mouseClicked(MouseEvent me){
System.out.println("Mouse Clicked ["+me.getX()+" , "+me.getY()+"]");
}
public void mousePressed(MouseEvent me){
System.out.println("Mouse Pressed ["+me.getX()+" , "+me.getY()+"]");
}
public void mouseReleased(MouseEvent me){
System.out.println("Mouse Released ["+me.getX()+" , "+me.getY()+"]");
}
public void mouseEntered(MouseEvent me){
System.out.println("Mouse Entered ["+me.getX()+" , "+me.getY()+"]");
}
public void mouseExited(MouseEvent me){
System.out.println("Mouse Exited ["+me.getX()+" , "+me.getY()+"]");
}
```

```
Class MyFrame extends Frame {
```

```
MyFrame(){
```

```
this.setVisible(true);
```

```
this.setSize(500,500);
```

```
this.setTitle("Mouse Events Example");
```

```

 this.setBackground(Color.green);
 this.addWindowListener(new WindowAdapter()
 {
 public void windowClosing(WindowEvent we)
 {
 System.exit(0);
 }
 });
 this.addMouseListener(new MouseListenerImpl());
}

class MouseEventsFor
{
 public String[] args)
 {
 Myframe mf = new Myframe();
 }
}

E O/P:- Mouse

```

Ex:-

```

public void keypressed(KeyEvent ke)
{
 S.O.P("Key Pressed:" + ke.getKeyChar());
}

public void keyreleased(KeyEvent ke)
{
 S.O.P("Key Released:");
}

```

**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Amerpet, Hyderabad.

Class Myframe extends frame

```
{
 Myframe()
 {
 f1.setVisible(true);
 f1.setSize(500,500);
 f1.setTitle("Key Event Example");
 f1.setBackground(Color.green);
 KeyListenerImpl kli = new KeyListenerImpl();
 f1.addKeyListener(kli);
 f1.addWindowListener(new WindowAdapter()
 {
 public void windowClosing(WindowEvent we)
 {
 System.exit(0);
 }
 });
 }

 class KeyEventEx
 {
 public void main(String[] args)
 {
 Myframe = new Myframe();
 }
 }
}
```

## Layout Mechanisms:-

The main intention of layout mechanisms is to arrange all the GUI components in the particular order.

There are five types of layout mechanisms in AWT.

- ① FlowLayout
- ② BorderLayout
- ③ GridLayout
- ④ GridBagConstraints
- ⑤ CardLayout

### ① FlowLayout:-

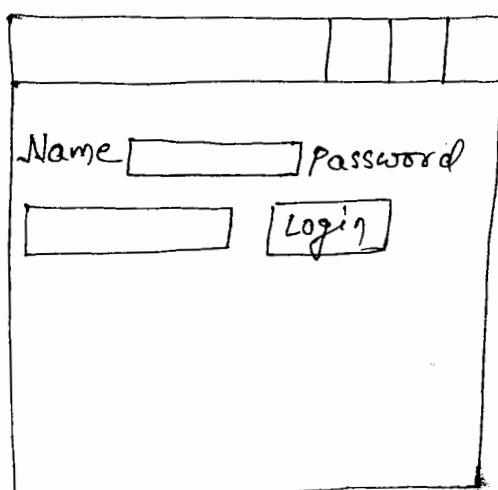
This layout mechanism is able to arrange all the GUI components in rows wise manner.

To represent this mechanism, AWT has provided a predefined class in the form of `java.awt.FlowLayout`.

To set this layout mechanism to frame we have to use the following method.

```
public void setLayout(XXXLayout l)
```

Ex:- `f.setLayout(new FlowLayout());`



## ② BorderLayout:-

If we want to represent all the GUI components along with borders of the frame then we have to use BorderLayout.

To represent BorderLayout, AWT has provided a predefined class in the form of `java.awt.BorderLayout`.

To set BorderLayout to frame we have to use the following instruction.

`f.setLayout(new BorderLayout());`

To represent locations in BorderLayout, BorderLayout class has provided the constants like EAST, WEST, NORTH, SOUTH, CENTER.

→ To add any GUI components in BorderLayout we have to use the following method.

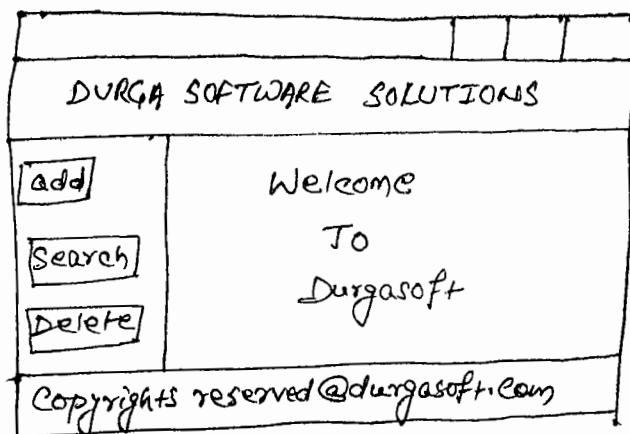
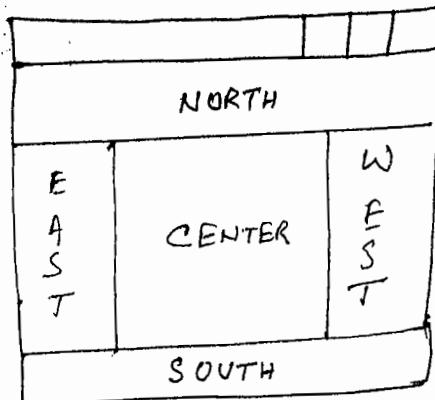
`public void add (int location, Component c)`

Eg:-

```
MenuPanel mp = new MenuPanel();
HeaderPanel hp = new HeaderPanel();
FooterPanel fp = new FooterPanel();
WelcomePanel wp = new WelcomePanel();
f.setLayout(new BorderLayout());
f.add(BorderLayout.EAST, mp);
f.add(BorderLayout.NORTH, hp);
f.add(BorderLayout.CENTER, wp);
f.add(BorderLayout.SOUTH, fp);
```

Z

P.T.O.  
Cont. --



### ③ Grid Layout:-

If we want to arrange all the GUI Components in the form of rows and columns then we have to use GridLayout.

To represent GridLayout, AWT has provided a predefined class in the form of `java.awt.GridLayout`.

To use GridLayout in GUI applications, we have to use the following code.

f. `setLayout(new GridLayout(4, 4));`

where 4,4 are no. of rows and columns.

→ In case of GridLayout, we should not allow empty grids between GUI components.

→ In GridLayout, all the GUI components must equal size.

∴

|   |   |   |   |
|---|---|---|---|
|   |   |   |   |
| 1 | 2 | 3 | + |
| 4 | 5 | 6 | - |
| 7 | 8 | 9 | * |
| 0 | . | = | / |

#### ④ GridBagLayout:-

This layout mechanism is almost all same as GridLayout, but in GridBagLayout, we are able to provide empty grids between GUI Components and GUI components sizes may be varied.

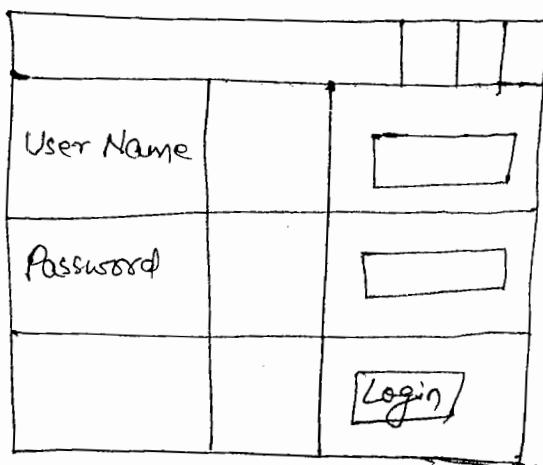
To represent GridBagLayout, AWT has provided a predefined class in the form of `java.awt.GridBagLayout`.

To set GridBagLayout to frame we have to use the following

`f.setLayout(new GridBagLayout(3,3));`

where 3,3 is no. of rows and no. of columns.

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.



#### ⑤ CardLayout:-

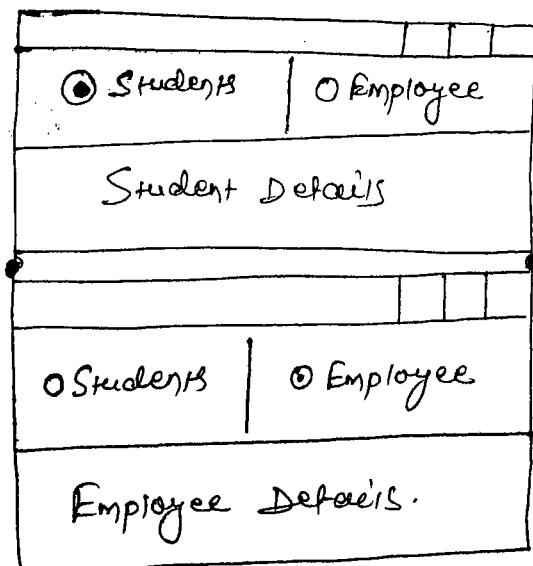
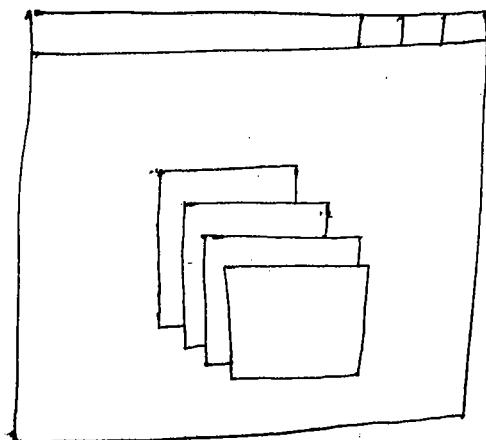
This layout mechanism is able to allow all the GUI Components in layers wise and in overlapped manner.

To represent this layout mechanism, AWT has provided a predefined ~~library~~ class in the form of `java.awt.CardLayout`.

If we want to use CardLayout mechanism in GUI applications then we have

`f.setLayout(new CardLayout(5));`

where 5 is no. of cards.



### Label :-

It is an output GUI component, it able to display a text data on frame.

To represent this GUI component AWT has provided predefined class in the form of

`java.awt.Label`

To Create Label Class objects we have to use the following Constructors.

`public Label()`

`public Label(String label)`

`public Label(String label, int alignment)`

### Button :-

It is an output GUI component, it able to raise an ~~ActionEvent~~ ActionEvent in order to perform an action in GUI Applications.

To represent Button in GUI Applications, AWT has provided a predefined class in the form of `java.awt.Button`.

If we want to create Button Class object in GUI Applications then we have to use the following Constructors.

```
public Button()
public Button(String label)
```

To set label to the button and to get label from button we have to use the following methods.

```
public void setLabel(String label)
public String getLabel()
```

If we want to get label of the clicked button among multiple buttons then we have to use the following method.

```
public String getActionCommand()
```

Ex:-

```
import java.awt.*;
import java.awt.event.*;

Class Myframe extends Frame implements ActionListener
{
 Button b1, b2, b3;
 Myframe()
 {
 f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 f.setSize(500, 500);
 f.setTitle("Button Example");
 f.setLayout(new FlowLayout());
 f.addWindowListener(new WindowAdapter()
 {
 public void windowClosing(WindowEvent we)
 {
 System.exit(0);
 }
 });
 b1 = new Button("Red");
 b2 = new Button("Green");
 b3 = new Button("Blue");
 }
}
```

```

b1.addActionListener(this);
b2.addActionListener(this);
b3.addActionListener(this);
b1.setBackground(Color.red);
b2.setBackground(Color.green);
b3.setBackground(Color.Blue);
this.add(b1);
this.add(b2);
this.add(b3);
}

public void actionPerformed(ActionEvent ae)
{
 String label = ae.getActionCommand();
 if (label.equals("Red"))
 {
 this.setBackground(Color.red);
 }
 if (label.equals("Green"))
 {
 this.setBackground(Color.green);
 }
 if (label.equals("Blue"))
 {
 this.setBackground(Color.blue);
 }
}

class ButtonEx {
 public static void main(String[] args) {
 Myframe mf = new Myframe();
 }
}

```

## TextField :-

It is an input GUI component, it able to allow a line of text data in order to submit that data to java program.

To represent TextField GUI Component AWT has provided a predefined class "java.awt.TextField".

To create TextField class object we have to use the following Constructors.

```
public TextField()
 (int size)
 (String def-message)
 (String def-message, int size)
```

To set data to text field and to get data from text field we have to use the following methods.

```
public void setText(String text-data)
public String getText()
```

## TextArea:-

It is an input GUI component, it able to allow more than one line of data in order to submit data to java application.

To represent TextArea in GUI applications, AWT has provided a predefined class in the form of "java.awt.TextArea".

To create TextArea class object we have to use the following Constructors.

```
public TextArea()
 (String def-message)
 (int rows, int cols)
 (String def-message, int rows, int cols)
```

(13)

To set data to Text Area and to get data from Text Area we have to use the following method.

```
public void setText(String data)
public String getText()
```

S

Note:- In AWT, there is no predefined class to represent password field, we have to convert text field into password field by hiding data in text field with a particular character.

To hide data in textfield with a particular character we have to use the following method.

```
public void setEchoChar(char c). I
```

Eg:-

```
import java.awt.*;
import java.awt.event.*;
Class Myframe extends frame implements ActionListener
{
 Label l1, l2, l3, l4, l5;
 Textfield tf1, tf2, tf3, tf4;
 TextArea ta;
 Button b;
 String uname=" ", upwd=" ", uemail=" ", umobile=" ", uaddr=" "
 Myframe()
 {
 this.setVisible(true);
 this.setSize(800, 500);
 this.setTitle("Textfield Example");
 this.setLayout(null);
 this.setBackground(Color.green);
 this.addWindowListener(new WindowAdapter()
 {
```

```

public void windowClosing(WindowEvent we)
{
 System.exit(0);
}

l1 = new Label("User Name");
l2 = new Label("Password");
l3 = new Label("User Email");
l4 = new Label("User Mobile");
l5 = new Label("User Address");

tf1 = new TextField(20);
tf2 = new TextField(20);
tf2.setEditable(true);
tf2.setEchoChar('*');
tf3 = new TextField(20);
tf4 = new TextField(20);
ta = new TextArea(10, 40);
b = new Button("Display");
b.addActionListener(this);

Font f = new Font("arial", Font.BOLD, 20);
l1.setFont(f);
l2.setFont(f);
l3.setFont(f);
l4.setFont(f);
l5.setFont(f);

tf1.setFont(f);
tf2.setFont(f);
tf3.setFont(f);
tf4.setFont(f);

```

**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

```

 t2.setfont(f);
 b.setfont(f);

 l1.setBounds(50, 100, 200, 30);
 tf1.setBounds(260, 100, 200, 30);
 l2.setBounds(50, 150, 200, 30);
 tf2.setBounds(260, 150, 200, 30);
 l3.setBounds(50, 200, 200, 30);
 tf3.setBounds(260, 200, 200, 30);
 l4.setBounds(50, 250, 200, 30);
 tf4.setBounds(260, 250, 200, 30);
 l5.setBounds(50, 300, 200, 30);
 ta.setBounds(260, 300, 200, 100);
 s.setBounds(50, 350, 100, 30);

 this.add(l1); this.add(l1);
 this.add(l2); this.add(tf2);
 this.add(l3); this.add(tf3);
 this.add(l4); this.add(tf4);
 this.add(l5); this.add(ta);
 this.add(s);
}

public void actionPerformed(ActionEvent ae)
{
 uname = tf1.getText();
 upwd = tf2.getText();
 uemail = tf3.getText();
 umobile = tf4.getText();
 uaddr = ta.getText();
 repaint();
}

```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

```
public void paint(Graphics g)
{
 Font f = new Font("arial", Font.BOLD, 30);
 g.setFont(f);

 g.drawString("User Name :" + cname, 500, 100);
 g.drawString("User Password :" + upwd, 500, 150);
 g.drawString("User Email :" + uemail, 500, 200);
 g.drawString("User Mobile :" + umobile, 500, 250);
 g.drawString("User Address :" + uaddr, 500, 300);
}
```

```
Class TextfieldEx {
```

```
PSVM(String[] args) {
```

```
Myframe mf = new Myframe();
```

```
}
```

### Checkbox:-

It is an input GUI Component, it able to allow to select an item.

To represent Checkbox GUI Component in GUI Applications AWT has provided a predefined class in the form of "java.awt.Checkbox"

To create Checkbox class Object, we have to use the following Constructors

```

public CheckBox()
public CheckBox (String label)
public CheckBox (String label, boolean state)
public CheckBox (String label, CheckBoxGroup cg)
public CheckBox (String label, CheckBoxGroup cg, boolean b)

```

To set label to CheckBox and to get label from CheckBox we have to use the following methods.

```

public void setLabel(String label)
public String getLabel()

```

To set a particular state to CheckBox and to get current state from CheckBox we have to use the following methods

```
public void setState(boolean b)
```

```
public String boolean getState()
```

Note:- In GUI Applications, no predefined class is provided by AWT to represent radio button. To prepare radiobutton in GUI applications we have to create checkbox class Object with checkboxgroup Object reference as parameter.

Ex:-

```

CheckBoxGroup cbg = new CheckBoxGroup();
CheckBox cb1 = new CheckBox("Male", cbg, false);
CheckBox cb2 = new CheckBox("Female", cbg, false);

```

Ex:-

```

import java.awt.*;
import java.awt.event.*;
class Myframe extends Frame implements ItemListener
{
 Label l1, l2;
 CheckBox cb1, cb2, cb3, cb4, cb5
}

```

```

 this.setVisible(true);
 this.setSize(500,500);
 this.setTitle("Qualifications");
 String qual = " ";
 String gender = " ";
 MyFrame();
 }

 this.setVisible(true);
 this.setSize(500,500);
 this.setTitle("Checkbox Example");
 this.setLayout(new FlowLayout());
 this.setBackground(Color.green);
 this.addWindowListener(new WindowAdapter()
 {
 public void windowClosing(WindowEvent we)
 {
 System.exit(0);
 }
 });
}

l1 = new Label("Qualifications");
l2 = new Label("User Gender");
cb1 = new Checkbox("BSC", false);
cb2 = new Checkbox("MCA", false);
cb3 = new Checkbox("PHD", false);
cb4 = new Checkbox("Male", cbg, false);
cb5 = new Checkbox("Female", cbg, false);
CheckboxGroup cbg = new CheckboxGroup();
cb4 = new Checkbox("Male", cbg, false);
cb5 = new Checkbox("Female", cbg, false);
Font f = new Font("Arial", Font.BOLD, 20);

```

```

l1.setfont(f)
l2.setfont
Cb1.setfont
Cb2.setfont
Cb3.setfont
Cb4.setfont
Cb5.setfont

```

```

Cb1.addItemListener(this);
Cb2.addItemListener(this);
Cb3.addItemListener(this);
Cb4.addItemListener(this);
Cb5.addItemListener(this);

```

```

this.add(l1);
this.add(Cb1);
this.add(Cb2);
this.add(Cb3);
this.add(l2);
this.add(Cb4);
this.add(Cb5);
}

```

```

public void ItemStateChanged(ItemEvent ie)
{
 if(Cb1.getState() == true)
 {
 qual = qual + Cb1.getLabel() + " ";
 }
 if(Cb2.getState() == true)
 {
 qual = qual + Cb2.getLabel() + " ";
 }
}

```

**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Amerpet, Hyderabad.

```
if (cb3.getState() == true)
{
 qual = qual + cb3.getLabel();
}
if (cb4.getState() == true)
{
 qual = qual + cb4.getLabel();
}
if (cb5.getState() == true)
{
 gender = cb5.getLabel();
}
repaint();
}

public void paint(Graphics g)
{
 Font f = new Font("arial", Font.BOLD, 30);
 g.setFont(f);
 g.drawString("Qualifications : " + qual, 50, 250);
 g.drawString("User Gender : " + gender, 50, 300);
 qual = "";
 gender = "";
}

class CheckBoxEx {
 public static void main (String [] args) {
 Myframe mf = new Myframe();
 }
}
```

### List:-

It is an input GUI component, it is able to display a list of items to select.

To represent List GUI Component in GUI Apps AWT has provided a predefined class `java.awt.List`.

To Create List class Object we have to use the following Constructors

```
public List()
public List(int size)
public List(int size, boolean state)
```

To add an item to list we have to use the following method.

```
public void add(String item)
```

To get the selected item from list we have to use the following method.

```
public String getSelectedItem()
```

To get more than one selected items from list we have to use the following method.

```
public String[] getSelectedItems()
```

### Choice:-

It is an input GUI component, it is able to allow to select Only one item.

To represent Choice GUI Component AWT has provided a predefined class in the form of `java.awt.Choice`.

To Create Choice class Object we have to use the following Constructors.

```
public Choice()
```

To add an item to Choice Box we have to use the following method.

public void add(String item)

To get Selected item from Choice Box we have to use the following method.

public String getSelectedItem()

Eg:-

```
import java.awt.*;
import java.awt.event.*;
```

Class Myframe extends frame implements ItemListener.

```
{ Label l1, l2;
List l;
```

```
Choice c;
```

```
String tech = " ";
```

```
String prof = " ";
```

```
Myframe()
```

```
{ this.setVisible(true);
this.setSize(500,500);
this.setTitle("List Box Example");
this.setLayout(new FlowLayout());
this.setBackground(Color.green);
this.addWindowListener(new WindowAdapter()
```

```
{ public void windowClosing(WindowEvent we)
```

```
System.exit(0);
```

```

 });
}

l1 = new label ("Technologies");
l2 = new label ("Profession");
l = new list (4, true);
l.add ("C");
l.add ("C++");
l.add ("JAVA");
l.add ("J2EE");

C = new Choice();
C.add ("Student");
C.add ("Business");
C.add ("Employee");
l.addItemListener (this);
C.addItemListener (this);

Font f = new Font ("arial", Font.BOLD, 20);
l1.setFont (f);
l2.setFont (f);
l.setFont (f);
C.setFont (f);

this.add (l1);
this.add (l2);
this.add (l2);
this.add (C);
}

public void ItemStateChanged (ItemEvent ee)
{
 String [] items = l.getSelectedItems ();
}

```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

```
for(int i=0; i<items; i++)
{
 tech = tech + items[i] + " ";
}
String prof = c.getSelectedItem();
repaint();
}
}
public void paint(Graphics g)
{
 Font f = new Font("Arial", Font.BOLD, 30);
 g.setFont(f);
 g.drawString("Technologies : " + tech, 50, 200);
 g.drawString("Profession : " + prof, 50, 250);
 tech = "";
}
}
class Lista
{
 public void main(String[] args)
 {
 Myframe mf = new Myframe();
 }
}
```

### Menu:-

Menu is able to display list of items to select an item.

To prepare menu in GUI applications then we have to use the predefined classes like MenuBar, Menu and MenuItem.

To prepareMenuBar we have to use the following constructor.

publicMenuBar()

Ex:- MenuBar mb=newMenuBar();

To set menuBar to frame we have to use the following method.

```
public void setMenuBar(MenuBar mb)
f.setMenuBar(mb);
```

To prepare menu we have to use the following constructor.

public menu(String name)

Ex:- menu m=new menu("file");

To add menu to menuBar we have to use the following method.

public void add(Menu m)

Ex:- mb.add(m);

To create MenuItem we have to use the following constructor.

public MenuItem(String name)

Ex:- MenuItem mi=new MenuItem("New");

To add MenuItem to menu we have to use the following method.

public void add(MenuItem mi)

Ex:- m.add(mi);

In GUI applications, MenuItem is able to raise ActionEvent and it will be handled by ActionListener, where if we want to get the selected MenuItem we have to use the following method from ActionEvent.

public String getActionCommand()

z

Eg:-

```
import java.awt.*;
import java.awt.event.*;
class Myframe extends Frame implements ActionListener
{
 MenuBar mb;
 Menu m;
 MenuItem mi1, mi2, mi3;
 String item = " ";
 Myframe()
 {
 this.setVisible(true);
 this.setSize(500, 500);
 this.setTitle("Menu Example");
 this.setBackground(Color.green);
 this.addWindowListener(new WindowAdapter()
 {
 public void windowClosing(WindowEvent we)
 {
 System.exit(0);
 }
 });
 mb = new MenuBar();
```

```

this. setMenuBar(mb);
m = new Menu("File");
mb.add(m);
mi1 = new MenuItem("New");
mi2 = new MenuItem("Open");
mi3 = new MenuItem("Save");
m.add(mi1);
m.add(mi2);
m.add(mi3);
mi1.addActionListener(this);
mi2.addActionListener(this);
mi3.addActionListener(this);
}

public void actionPerformed(ActionEvent ae)
{
 item=ae.getActionCommand();
 repaint();
}

public void paint(Graphics g)
{
 Font f = new Font("Arial", Font.BOLD, 30);
 g.setFont(f);
 g.drawString("Selected Item : " + item, 100, 250);
}

class MenuEx {
 public static void main(String[] args) {
 Myframe mf = new Myframe();
 }
}

```

## ScrollBar:-

This GUI Component can be used to move frame to top to bottom or left to right on the basis of the type of ScrollBar which we used.

To represent ScrollBar GUI component AWT has provided a predefined class in the form of `java.awt.ScrollBar`. To create ScrollBar class object we have to use the following constructor.

`public ScrollBar( int type )`

where type may be VERTICAL or HORIZONTAL Constants from ScrollBar class.

To get position of the Scrollbar we have to use the following method.

`public int getValue()`

Ex:-

```
import java.awt.*;
import java.awt.event.*;
```

Class Myframe extends Frame implements AdjustmentListener

```
{
 Scrollbar sb;
 int position;
```

```
 Myframe()
 {
```

```
 this.setVisible(true);
 this.setSize(500,500);
 this.setTitle("Scrollbar Example");
 this.setBackground(Color.green);
```

```

this.setLayout(new BorderLayout());
this.addWindowListener(new WindowAdapter()
{
 public void windowClosing(WindowEvent we)
 {
 System.exit(0);
 }
});

sb=new Scrollbar(Scrollbar.VERTICAL);
sb.setAdjustmentListener(this);
this.add(BorderLayout.EAST,sb);

public void adjustmentValueChanged(AdjustmentEvent ae)
{
 position=sb.getValue();
 repaint();
}

public void paint(Graphics g)
{
 Font f=new Font("arial",Font.BOLD,30);
 g.setFont(f);
 g.drawString("Position : "+position,100,250);
}

class ScrollBar/
PSVM(String[] args)
{
 Myframe mf=new Myframe();
}

```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

## Applets:-

Applet is a container, it able to include some other GUI components.

Q. What is the difference between Application and Applet?

→ Application is a java program, where main() method is called to executed and it will be executed by using JVM.

An Applet is a Java program, it will not have main() method, it will have some lifecycle methods to execute and it will be executed by either web browsers or by using Appletviewer.

If we want to use Applets in GUI applications then we have to use the following steps.

- following steps.

  - ① Create Applet class.
  - ② Create Applet Configuration file.
  - ③ Execute Applet.

## ① Create Applet Class:-

To create Applet class we have to declare an user defined class, it must be extended from `java.applet.Applet` class, where we have to override Applet lifecycle methods.

```
public class MyApplet extends Applet
{
 — Overide init(), start(), stop(), destroy() —
}
```

## ② Prepare Applet Configuration file:-

To prepare Applet Configuration file we have to declare an HTML file, where we have to use the following tag.

(22)

```
<applet code = "___" width = "___" height = "___">
</applet>
```

Where "code" attribute will take the name and location of the Applet class.

Where "width" and "height" attributes can be used to specify Applet Size.

Ex:- MyApplet.htm

```
<applet cod = "MyApplet" width = "500" height = "500">
</applet>
```

### ③ Execute Applet:-

There are two ways to execute Applets.

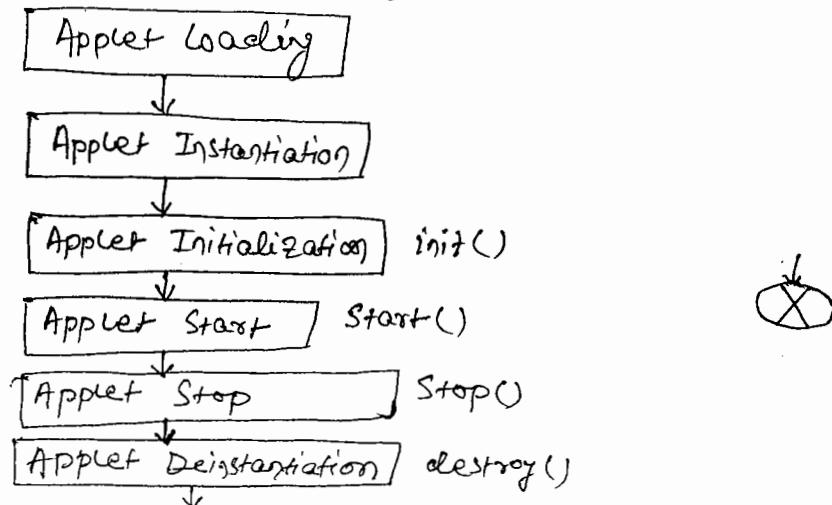
- ① By using web browsers
- ② By using appletViewer tool.

Applet was introduced by Java along with JDK1.0 version, so it is not supported by latest versions of browsers.

If we want to execute applet by using appletViewer then we have to use the following command.

D:\apps>appletviewer MyApplet.htm).

When we execute Applet the AppletViewer will execute applet by using the following lifecycle actions.



Edit

logoApplet.java

```
import java.awt.*;
import java.applet.*;
public class logoApplet extends Applet
{
 public void paint(Graphics g)
 {
 font f = new Font("arial", Font.BOLD, 40);
 g.setFont(f);
 this.setBackground(Color.green);
 this.setForeground(Color.red);
 g.drawString("DURGA SOFTWARE SOLUTIONS", 100, 100);
 }
}
```

LogoApplet.html

```
<applet code = "LogoApplet" width = "700" height = "400">
</applet>
```

D:\javaf\gui> appletviewer LogoApplet.html

finish applet

Swing:-

Initially, SUN Microsystems has provided AWT to prepare GUI apps, but AWT GUI Components will provide less performance in GUI applications.

In this context, Netscape Communications has provided another toolkit to design GUI apps. in the form of JFC (Internet Foundation Classes), but JFC provided GUI components are also for improving GUI applications performance.

In this situation, both sun microsystems and netscape communications has provided ~~at~~ a common toolkit to prepare GUI applications. that is JFC (Java Foundation Classes).

JFC has provided the following features to design GUI apps.

- 3D Color Monitor
- Clipboard Support
- Copy and paste Capabilities
- Drag and Drop Capabilities
- SWING

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Q: What are differences between AWT and SWING?

① AWT GUI Components are platform dependent.

SWING GUI Components are platform independent.

② AWT is able to provide fixed look and feel for its GUI components which is supported by the Operating System.

SWING is able to provide pluggable look and feel to its GUI components irrespective of the Operating System which we used.

③ AWT GUI Components are heavy weight GUI Components.  
SWING GUI components are light weight GUI components.

④ AWT GUI Components are peer components, for every GUI Component in GUI application a separate duplicate component will be created at Operating System level.

SWING GUI Components are not peer components, duplicate components are not created at Operating System level.

⑤ AWT has provided only the basic GUI components. SWING has provided the most advanced GUI components like Color Chooser, Separators, trees, ...

⑥ AWT has provided less no. of GUI components, SWING has provided 4 times <sup>greater</sup> no. of components when compared with AWT components.

⑦ AWT GUI Components are not supporting "Tool Tip Text" message.

~~All the AWT GUI components are not supporting Tool Tip Text message.~~  
~~But almost all the SWING GUI components are supporting Tool Tip Text message.~~

To set Tool Tip Text message to GUI components in SWING we have to use the following method.

```
public void setToolTipText(String msg)
```

⑧ In AWT to close the frames, we have to use WindowListener implementations or WindowAdapter abstract class or anonymous inner class.

In case of SWING, to close frame we have to use the following instruction:

```
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

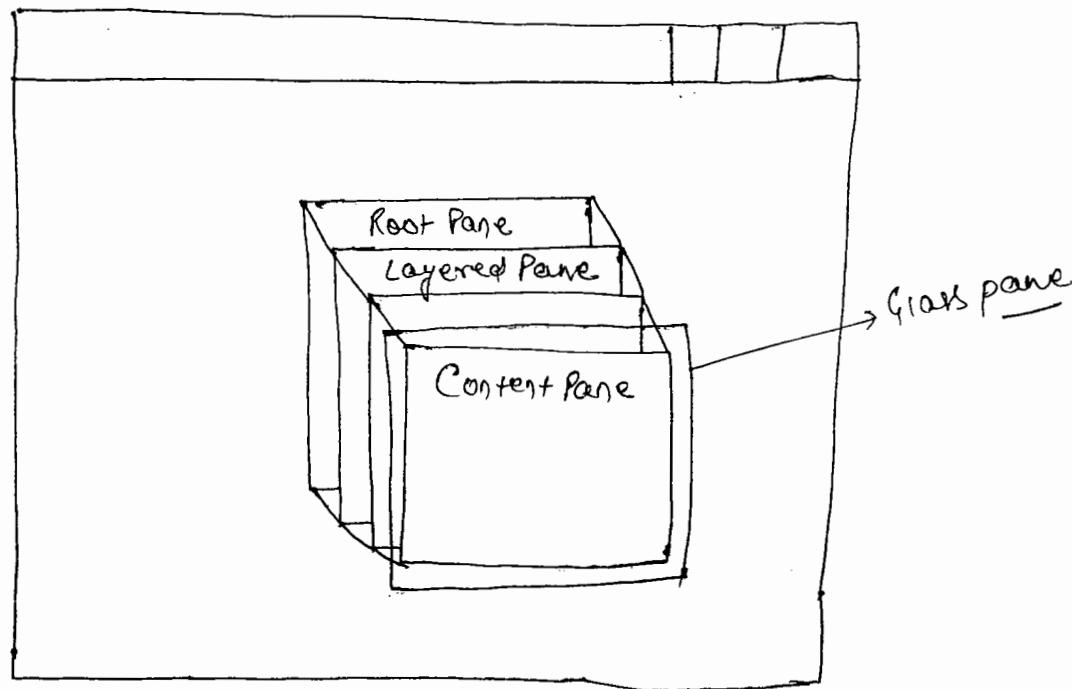
⑨ In case of AWT, we will add all the GUI components directly to frame.

In case of SWING, we will not add GUI components to frame directly, we will add GUI components to panes managed by frame.

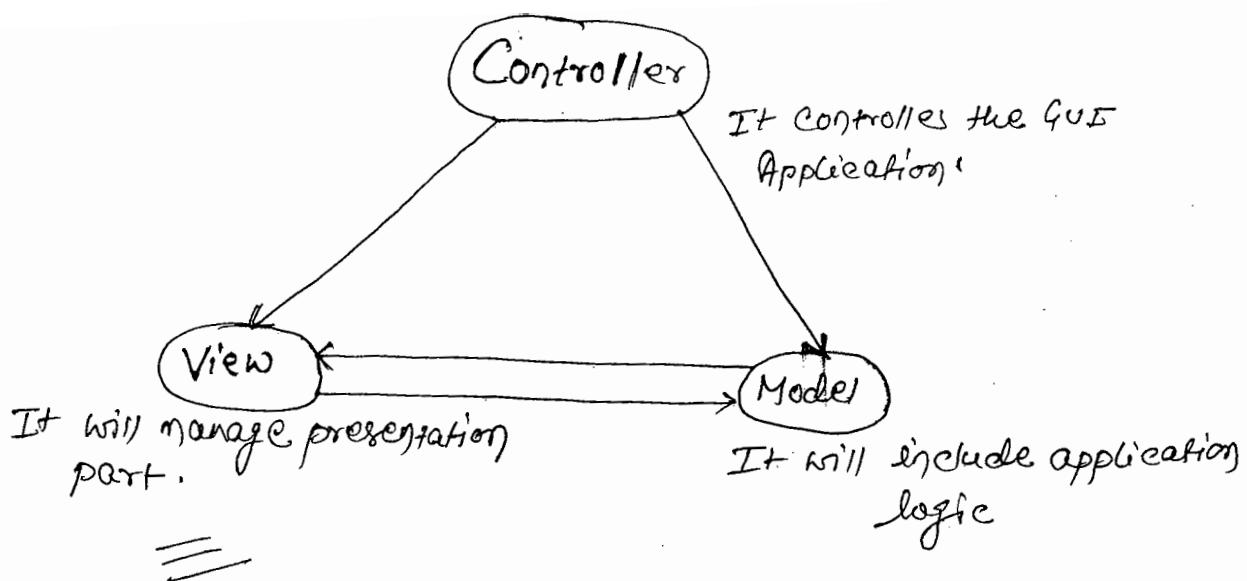
In SWING, there are four types of pane.

- ① Root pane:- It will include LayeredPane and all the GUI Components references.
- ② LayeredPane:- It will include all the GUI Components in the form of layers. Hence is the best suitable component for layered pane.
- ③ Content pane:- It is a general pane, it will include all the GUI components which are provided.
- ④ Glass Pane:- It will cover all the GUI components of Content pane for only viewing not for updations. If we want to get Content pane in GUI applications then we have to use

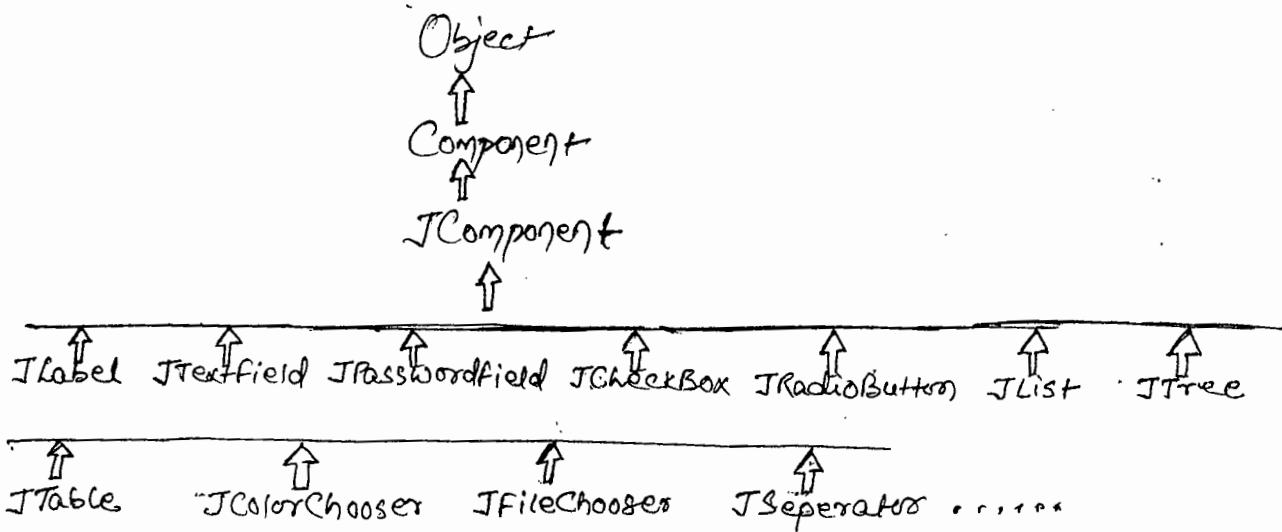
public Container getContentPane()



- ⑩ AWT Applications are not following MVC Arch internally.  
SWING application will follow MVC Arch internally.



To prepare GUI applications, SWING has provided the following predefined library in the form of javax.swing package.



### JColorChooser:-

This GUI component is able to display a list of colors to select. To represent ColorChooser, SWING has provided a predefined class in the form of javax.swing.JColorChooser.

To Create JColorChooser Class Object we have to use the following constructor.

```
public JColorChooser()
```

If we want to get the selected color from ColorChooser then we have to use the following method.

public Color getColor()

ColorChooser is able to rise ChangeEvent and it would be handled by ChangeListener by executing the following method.

public void stateChanged(ChangeEvent e)

If we want to add ChangeListener to TColorChooser then first we have to get SelectionModel then we have to add ChangeListener.

To get SelectionModel we have to use the following method.

public SelectionModel getSelectionModel()

Ex:- ce.getSelectionModel().addChangeListener(this);

### JFileChooser:-

This GUI Component is able to display file dialog to select a file and to get the name and location of the file.

To represent this GUI Component, SWING has provided a predefined class in the form of javax.swing.JFileChooser.

To Create JFileChooser Class Object we have to use the following constructor.

public JFileChooser()

To get the selected file in the form of File class object we have to use the following method.

public File getSelectedFile()

JFileChooser GUI Component is able to rise ActionEvent and it will be handled by ActionListener.

## JTable:-

This GUI component is able to display data in the form of table representation.

To represent this GUI component, swing has provided a predefined class in the form of javax.swing.JTable.

To create JTable class object we have to use the following constructor.

public JTable(Object[][] body, Object[] header)

To get table header separately in the form of javax.swing.JTableHeader object we have to use the following method.

public JTableHeader getTableHeader()

Swing finished

DURGASOFT

```
1 ColorChooserEx.java
2 =====
3 import javax.swing.*;
4 import javax.swing.event.*;
5 import java.awt.*;
6 class ColorsFrame extends JFrame implements ChangeListener
7 {
8 JColorChooser cc;
9 Container c;
10 ColorsFrame()
11 {
12 this.setVisible(true);
13 this.setSize(700,500);
14 this.setTitle("Color Chooser Example");
15 this.setLayout(new FlowLayout());
16 this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17 c= getContentPane();
18 cc=new JColorChooser();
19 cc.getSelectionModel().addChangeListener(this);
20 c.add(cc);
21 }
22 public void stateChanged(ChangeEvent ce)
23 {
24 Color clr=cc.getColor();
25 JFrame f=new JFrame();
26 f.setVisible(true);
27 f.setSize(300,300);
28 f.getContentPane().setBackground(clr);
29 }
30 }
31 class ColorChooserEx
32 {
33 public static void main(String[] args)
34 {
35 ColorsFrame f=new ColorsFrame();
36 }
37 }
38 =====
39 FileChooserEx.java
40 =====
41 import javax.swing.*;
42 import java.awt.*;
43 import java.awt.event.*;
44 import java.io.*;
45 class FileChooserFrame extends JFrame implements ActionListener
46 {
47 JLabel l;
48 JTextField tf;
49 JButton b;
50 Container c;
51 JFileChooser fc;
```



SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

**DURGASOFT**

---

```
53 FileChooserFrame()
54 {
55 this.setVisible(true);
56 this.setSize(700,500);
57 this.setLayout(new FlowLayout());
58 this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
59 c=getContentPane();
60 c.setBackground(Color.green);
61
62 l=new JLabel("Select File");
63 tf=new JTextField(20);
64 b=new JButton("Browse");
65 b.addActionListener(this);
66
67 Font f=new Font("arial",Font.BOLD,20);
68 l.setFont(f);
69 tf.setFont(f);
70 b.setFont(f);
71
72 c.add(l);c.add(tf);c.add(b);
73 }
74 public void actionPerformed(ActionEvent ae)
75 {
76 class FileDialogFrame extends JFrame implements ActionListener
77 {
78 FileDialogFrame()
79 {
80 this.setVisible(true);
81 this.setSize(600,500);
82 Container con=getContentPane();
83 con.setLayout(new FlowLayout());
84 fc=new JFileChooser();
85 fc.addActionListener(this);
86 con.add(fc);
87 }
88 public void actionPerformed(ActionEvent ae)
89 {
90 File f=fc.getSelectedFile();
91 String path=f.getAbsolutePath();
92 tf.setText(path);
93 this.setVisible(false);
94 }
95 }
96 FileDialogFrame fd=new FileDialogFrame();
97 }
98 }
99 class FileChooserEx
100 {
101 public static void main(String[] args)
102 {
103 FileChooserFrame fcf=new FileChooserFrame();
104 }
}
```





```

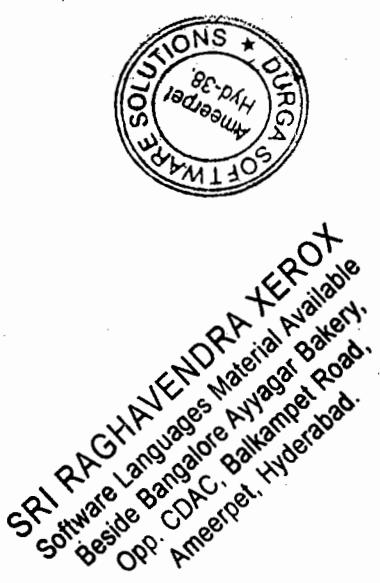
105 }
106 =====
107 SwingEx.java
108 =====
109 import java.awt.*;
110 import java.awt.event.*;
111 import javax.swing.*;
112 class RegistrationFrame extends JFrame implements ActionListener
113 {
114 JLabel l0,l1,l2,l3,l4,l5,l6,l7;
115 JTextField tf1;
116 JPasswordField pf1;
117 JCheckBox cb1,cb2,cb3;
118 JRadioButton rb1,rb2;
119 JList l;
120 JComboBox cb;
121 JTextArea ta;
122 JButton b;
123 Container c;
124
125
126 String uname="",upwd="",uqual="",ugender="",ulocations="",uprofession="",uaddr="";
127 RegistrationFrame()
128 {
129 this.setVisible(true);
130 this.setSize(600,700);
131 this.setTitle("Swing Example");
132 this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
133 c=getContentPane();
134 c.setLayout(null);
135 c.setBackground(Color.green);
136
137 l0=new JLabel("Registration Frame");
138 l1=new JLabel("User Name");
139 l1.setToolTipText("label");
140 l2=new JLabel("Password");
141 l2.setToolTipText("Label");
142 l3=new JLabel("Qualifications");
143 l4=new JLabel("Gender");
144 l5=new JLabel("Locations");
145 l6=new JLabel("Profession");
146 l7=new JLabel("Address");
147
148 tf1=new JTextField(20);
149 tf1.setToolTipText("Text Field");
150 pf1=new JPasswordField(20);
151 pf1.setToolTipText("Password Field");
152 cb1=new JCheckBox("BSC",false);
153 cb2=new JCheckBox("MCA",false);
154 cb3=new JCheckBox("PHD",false);
155 rb1=new JRadioButton("Male",false);
156 rb2=new JRadioButton("Female",false);

```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## DURGASOFT

```
157 ButtonGroup bg=new ButtonGroup();
158 bg.add(rb1);
159 bg.add(rb2);
160 String[] items1={"Hyderabad","Chennai","Pune","Delhi"};
161 l=new JList(items1);
162 String[] items2={"Employee","Business","Student"};
163 cb=new JComboBox(items2);
164 ta=new JTextArea(5,15);
165 b=new JButton("Registration");
166 b.addActionListener(this);
167
168 Font f1=new Font("arial",Font.BOLD+Font.ITALIC,40);
169 l0.setFont(f1);
170 Font f=new Font("arial",Font.BOLD,20);
171
172 l1.setFont(f);
173 l2.setFont(f);
174 l3.setFont(f);
175 l4.setFont(f);
176 l5.setFont(f);
177 l6.setFont(f);
178 l7.setFont(f);
179 tf1.setFont(f);
180 pf1.setFont(f);
181 cb1.setFont(f);
182 cb2.setFont(f);
183 cb3.setFont(f);
184 rb1.setFont(f);
185 rb2.setFont(f);
186 l.setFont(f);
187 cb.setFont(f);
188 ta.setFont(f);
189 b.setFont(f);
190
191 l0.setBounds(50,40,400,50);
192 l1.setBounds(50,100,200,20);
193 tf1.setBounds(260,100,200,30);
194 l2.setBounds(50,150,200,20);
195 pf1.setBounds(260,150,200,30);
196 l3.setBounds(50,200,200,20);
197 cb1.setBounds(260,200,100,30);
198 cb2.setBounds(370,200,100,30);
199 cb3.setBounds(480,200,100,30);
200 l4.setBounds(50,250,200,20);
201 rb1.setBounds(260,250,100,30);
202 rb2.setBounds(370,250,100,30);
203 l5.setBounds(50,300,200,20);
204 l.setFont(260,300,200,120);
205 l6.setBounds(50,450,200,20);
206 cb.setBounds(260,450,200,30);
207 l7.setBounds(50,500,200,20);
208 ta.setBounds(260,500,200,120);
```





```

209 b.setBounds(50,550,150,40);
210 c.add(l0);
211 c.add(l1);
212 c.add(tf1);
213 c.add(l2);
214 c.add(pf1);
215 c.add(l3);
216 c.add(cb1);
217 c.add(cb2);
218 c.add(cb3);
219 c.add(l4);
220 c.add(rb1);
221 c.add(rb2);
222 c.add(l5);
223 c.add(l1);
224 c.add(l6);
225 c.add(cb);
226 c.add(l7);
227 c.add(ta);
228 c.add(b);
229
230 }
231 public void actionPerformed(ActionEvent ae)
232 {
233 uname=tf1.getText();
234 upwd=pf1.getText();
235 if(cb1.isSelected()==true)
236 {
237 uqual=uqual+cb1.getLabel()+" ";
238 }
239 if(cb2.isSelected()==true)
240 {
241 uqual=uqual+cb2.getLabel()+" ";
242 }
243 if(cb3.isSelected()==true)
244 {
245 uqual=uqual+cb3.getLabel()+" ";
246 }
247 if(rb1.isSelected()==true)
248 {
249 ugender=rb1.getLabel();
250 }
251 if(rb2.isSelected()==true)
252 {
253 ugender=rb2.getLabel();
254 }
255 Object[] items=l.getSelectedValues();
256 for(int i=0;i<items.length;i++)
257 {
258 ulocations=ulocations+items[i]+" ";
259 }
260 uprofession=(String)cb.getSelectedItem();

```

**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

## DURGASOFT

```
261 uaddr=ta.getText();
262
263 class DisplayFrame extends JFrame
264 {
265 DisplayFrame()
266 {
267 this.setVisible(true);
268 this.setSize(800,500);
269 this.setTitle("Display Frame");
270
271 }
272 public void paint(Graphics g)
273 {
274 Font f=new Font("arial",Font.BOLD,30);
275 g.setFont(f);
276 g.drawString("User Name :"+uname,50,100);
277 g.drawString("Password :"+upwd,50,150);
278 g.drawString("Qualifications :"+uqual,50,200);
279 g.drawString("User Gender :"+ugender,50,250);
280 g.drawString("Locations :"+ulocations,50,300);
281 g.drawString("Profession :"+uprofession,50,350);
282 g.drawString("User Address :"+uaddr,50,400);
283 }
284 }
285 DisplayFrame df=new DisplayFrame();
286 }
287 }
288 class SwingEx
289 {
290 public static void main(String[] args)
291 {
292 RegistrationFrame rf=new RegistrationFrame();
293 }
294 }
=====
296 TableEx.java
=====
298 import javax.swing.*;
299 import javax.swing.table.*;
300 import java.awt.*;
301 class TableEx
302 {
303 public static void main(String[] args) throws Exception
304 {
305 JFrame f=new JFrame();
306 f.setVisible(true);
307 f.setSize(500,500);
308 f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
309 Container c=f.getContentPane();
310 c.setLayout(new BorderLayout());
311 String[] header={"ENO","ENAME","ESAL","EADDR"};
312 String[][]
```

two dimensional array.



**DURGASOFT**

```
body={{"111","AAA","5000","Hyd"}, {"222","BBB","6000","Sec"}, {"333","CCC","700
0","VJA"}, {"444","DDD","8000","Hyd"}};
JTable t=new JTable(body,header);
JTableHeader th=t.getTableHeader();
c.add(BorderLayout.NORTH,th);
c.add(BorderLayout.CENTER,t);
317 }
318 }
319
```

*Ref 19/10/2015*



**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

2011-03-20  
Wang, C., et al.  
"A New Type of  
"Bridging" Cell  
in the Human  
Spinal Cord"  
J. Neurosci., 31(12),  
4322-4332

## JVM Arch :-

Virtual Machine is a software simulation of a physical machine and it will perform the operations like physical machine.

Eg:- JVM (Java Virtual Machine) for java programs.

CLR (Common Language Runner) for .NET Application.

KVM (Kernel Virtual Machine) for Linux based cmd

PVM (Parrot Virtual Machine) for PERL Scripting.

- - - - -  
- - - - -  
- - - - -

JVM:- The main purpose of JVM is to translate bytecode from neutral nature to local Operating System representations [Native code] and executing java programs.

JVM is having mainly the following five components.

- ① Class loader Sub System
- ② Memory Management System
- ③ Execution Engine.
- ④
- ⑤ Java Native Library.

### ① Class Loader Sub system:-

The main purpose of class loader sub system is to load the classes bytecode to the memory and performing all the required actions in java programs while loading classes bytecode to the memory.

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

- ① Loading
- ② Linking
- ③ Initialization.

### ① Loading:-

The main purpose of loading phases to load classes bytecode to the memory. To load classes bytecode to the memory, JVM will use the following three class loaders -

- ① BootStrap class loader
- ② Extension class loader
- ③ Application class loader

### ① BootStrap Class Loader:-

Where BootStrap class/loader will load all the predefined classes bytecode to the memory by searching for them at BootStrap location.

→ C:\Java\jdk1.7.0\jre\lib\rt.jar" file. BootStrap class loader is prepared in java, it was prepared in some other native language. BootStrap class loader is acting as parent to extension class.

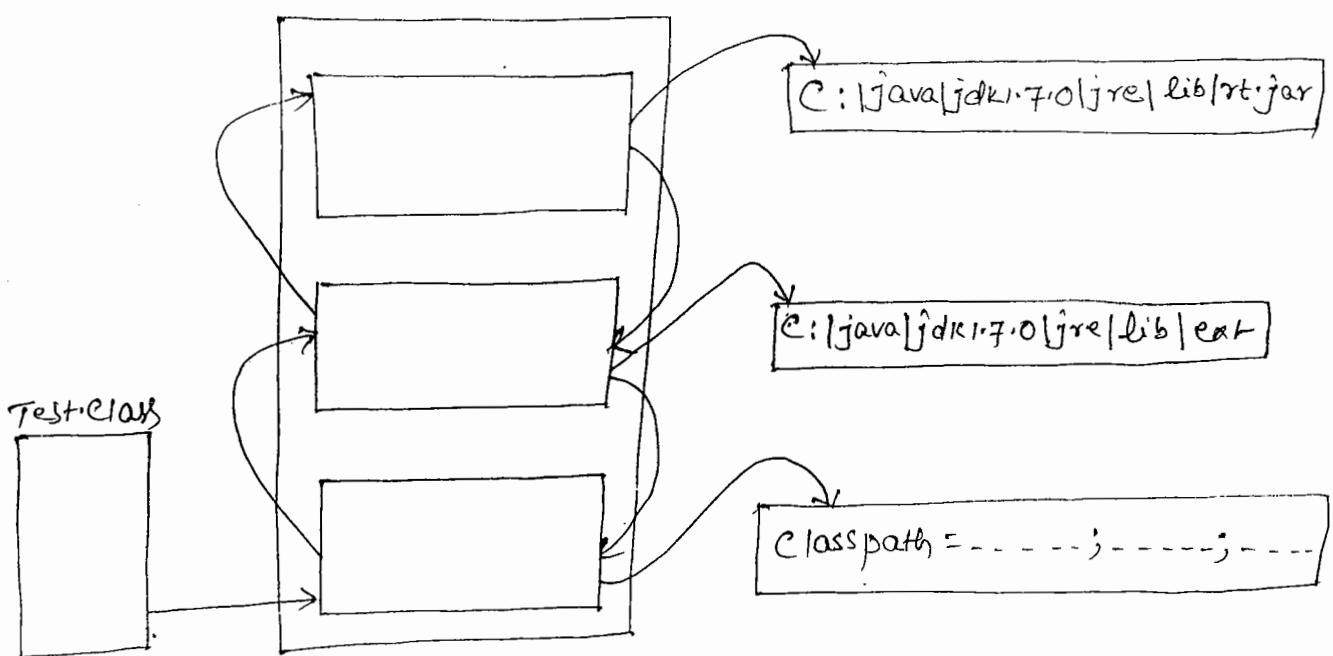
### ② Extension

Where extension class loader will load the classes and interfaces bytecode to the memory provided by third party libraries specified at ext location.

That is "C:\Java\jdk1.7.0\jre\lib\ext". This class loader was defined in java in the form of "Sun.misc.Launcher\$ExtClassLoader".

Where Application Class loader will load the classes and interfaces bytecode to the memory by searching for them at the locations referred by "classpath" environment variable. This classloader was defined in java in the form of "sun.misc.Launcher\$AppClassLoader".

When we execute java program, class loader system will bypass classes loading responsibility to Application Class Loader initially, where Application Class Loader <sup>will</sup> bypass class loading responsibility to Extension class loader, where Extension Class Loader will bypass class loading responsibility to Bootstrap class loader, where Bootstrap class loader will start class loading process and it will be terminated with Application Class Loader.



### Linking:-

The main purpose of linking is to provide links

between all the classes byte code in order to execute  
Linking phase is having mainly the following three phases.

### ① Verify:-

It will check whether byte code formations are proper or not, it will check whether class files are generated from right compilers or not....

### ② Prepare:-

This phase will assign memory for static variables in method area and it will provide default values to static variables.

### ③ Resolve:-

This phase will replace all the symbolic names with their actual references in main class byte code.

### ④ Initialization:-

This phase will initialize the static variables with their actual values and this phase will execute static block of the respective classes at the time of loading byte code.

## Memory Management System:-

The main purpose of memory management system is to store classes byte code and to store application data in the form of objects.

There are five types of memories in JVM.

- ① Method Area
- ② Heap Memory
- ③ Stack Memory
- ④ System PC Registers
- ⑤ Native Method Stacks.

## ① Method Area:-

The main purpose of Method Area is to store classes byte code, Constant pool areas data, static variables data.

→ Method Area will be created at the time of JVM startup.

→ Method Area is common memory for all the threads which are running in our application.

## ② Heap Memory:-

This memory is created at the time of JVM startup.

This memory is common memory for all the threads which are running in our application.

This memory is able to store all the entities data in the form of Objects and metadata of the classes which are loaded in method area in the form of `java.lang.Class` object.

In java applications, we are able to get heap memory details by using `java.lang.Runtime` class.

To get Runtime Class object we have to use the following static factory method.

`public static Runtime getRuntime()`

To get total initial memory which is assigned to heap then we have to use the following method.

`public long totalMemory()`

To get Max heap memory which JVM is able to assign for Heap we have to use the following method.

`public long maxMemory()`

To get free memory in heap we have to use the following method.

`public long freeMemory()`

$$\boxed{\text{Consumed Memory} = \text{Total Memory} - \text{Free Memory}}$$

Ex:-

```

Class Test {
 public static void main(String[] args) {
 Runtime rt = Runtime.getRuntime();
 System.out.println("Total Memory :" + rt.totalMemory());
 System.out.println("Free Memory :" + rt.freeMemory());
 System.out.println("Consumed Memory :" + rt.maxMemory());
 System.out.println("Consumed Memory :" + (rt.totalMemory() - rt.freeMemory()));
 }
}

```

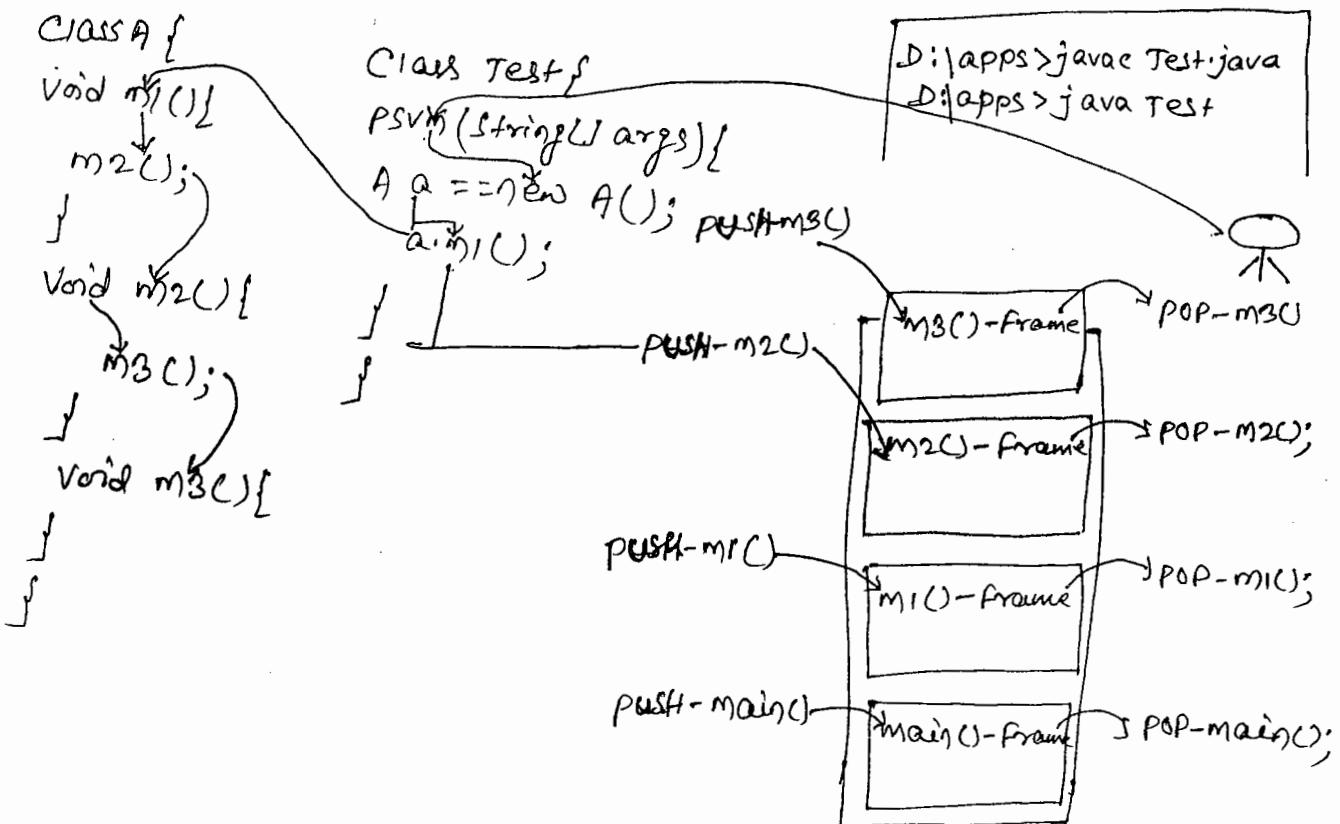
### ③ Stack Memory:-

- This memory is created whenever we are creating threads in java applications, not at the time of JVM Startup.
- This memory is not common memory to all the threads, this memory is specific to the threads.
- In java applications, when a thread is created then JVM will create a stack in stack memory, this stack is able to allow all the methods respective frames representing method details like name of the method, parameters of the method, local variables of the method and return value of the method when the respective thread access a method.

Z

(29)

In the above Context, when method execution is completed then JVM will pop the method respective frame from the stack.



#### ④ PC Register:-

It is a Register, it able to store address location of the instruction which is executed by the JVM at present.

#### ⑤ Native Method Stack:-

This memory is same as stack memory but it will be used for Native Methods Execution.



### ③ Execution Engine:-

The main purpose of Execution Engine is to execute java programs by translating neutral bytecode to the the native code.

In Execution Engine, there are three Components mainly.

- ① Interpreter
- ② JIT Compiler
- ③ Garbage Collector.

#### ① Interpreter:-

The main job of the Interpreter in Execution Engine is to translate neutral bytecode into local operating systems representations [native code] and executing that native code.

#### ② JIT Compiler/Just-In-Time Compiler]

In execution engine there is a problem with Interpreter, that is, if any method encountered by interpreter frequently then interpreter will translate that method neutral bytecode to native code and executes that native code every time, this approach will reduce JVM performance.

In the above context, to improve JVM performance, Sun Microsystems has provided JIT Compiler in execution engine.

→ JIT Compiler will manage a separate count for each and every method executed by the Interpreter and JIT Compiler will increment count value as per the same methods execution, if any method count value reached to its max value [Threshold value] then JIT Compiler will translate that method permanently to native code and given to Interpreter.

When it require without translating every time.

In JITCompiler, Count values and Threshold values are managed by "Profiler".

### (3) Garbage Collector:-

The main purpose of Garbage collector is to destroy objects whenever memory is filled with the objects completely.

In java applications, we can destroy objects explicitly by accessing Garbage Collector.

- ① Nullify Object reference
- ② Access System.gc() method.



When we access System.gc() method then Garbage collector will be activated, Garbage collector will destroy the nullified object, but just before destroying object, Garbage collector will access finalize() method in order to give final information.

Ex:-

```
Class A {
 A() {
 System.out.println("Object Creating");
 }
 public void finalize() {
 System.out.println("Object Destroying");
 }
}
```

```
Class Test {
 public static void main(String[] args) {
 A a = new A();
 a = null;
 System.gc();
 }
}
```

#### ④ Java Native Interface:-

It is an interface existed in between execution engine and java native library, it will translate native method calls from java to native language and the return values of native methods from native language to java programming language.

#### ⑤ Java Native Library:-

Java native library is a collection native methods, which are declared in java and implemented in non-java programming languages.

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## Exception Handling

29/10/2015

(20)

(31)

### Exception Handling:-

Q: What is the difference between error and exception?

→ Error is a problem, for which we are unable to provide solution programmatically.

Ex:- JVM Internal problem ✓

Insufficient Main Memory ✓

Unavailability of IO Components. ✓

-----

→ Exception is a problem, for which we are able to provide solution programmatically.

Ex:- ArrayIndexOutOfBoundsException ✓

ArithmaticException ✓

NullPointerException. ✓

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

### Exception:-

Exception is an unexpected event occurred at runtime provided by the users while entering dynamic input to the java applications, provided by the database engine while executing sql queries in JDBC Applications, provided by the network or Remote machine while establish the connection between Local Machine and Remote Machine Causes abnormal termination to the application.

→ In java applications, there are two types of terminations.

#### ① Smooth Termination:-

It is a termination to the java applications at end of the program.

#### ② Abnormal Termination:-

It is a termination to the java applications in middle of the program.

In java applications, Exceptions provided abnormal terminations may crash local Operating System or may provide hangs situation to the network based applications.

To Overcome all the above problems we have to handle exceptions properly, to handle exceptions we have to use a set of mechanisms called as Exception Handling Mechanisms.

→ The main Objective of Exception Handling Mechanisms is to handle exceptions in order to provide smooth terminations to the java applications.

→ Java is a robust programming language, because,

① Java is having very good memory management system in the form of Heap Memory Management System, it is a dynamic memory management system, it allocates and deallocates memory for the objects at runtime.

② Java is having very good exception handling mechanisms, because, Java has provided very good predefined library to represent all the Exception situations.

→ There are two types of exceptions in java applications:

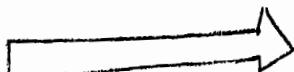
① Predefined Exceptions

② User-defined Exceptions.

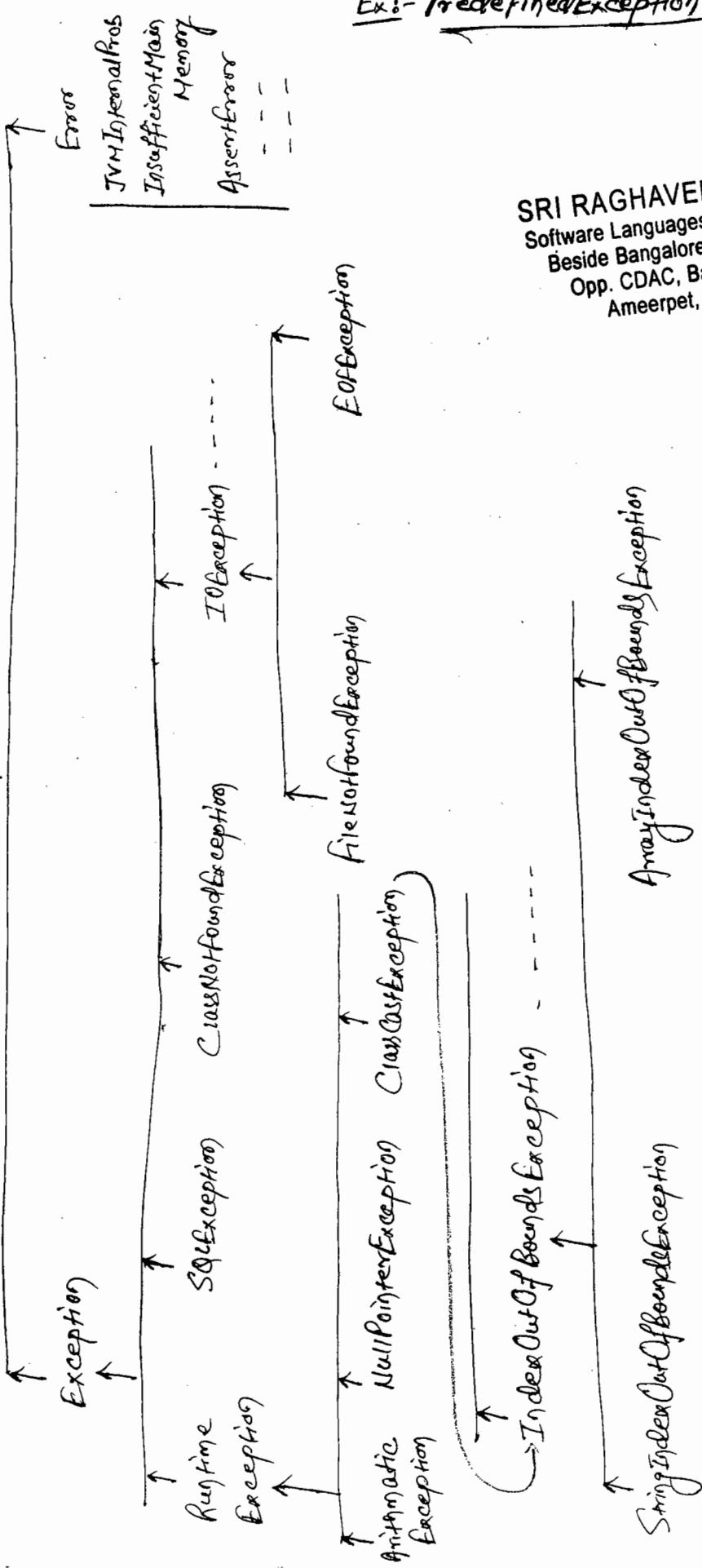
① Predefined Exceptions:-

These exceptions are provided by java programming language along with java software.

Ex:-



Object  
ThrowAsse



Ex:- PredefinedException

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

(32)

(31)

→ There are two types of Predefined exceptions

- (1) Checked Exceptions
- (2) Unchecked Exceptions.

Q: What is the diff. between Checked Exceptions and Unchecked Exceptions?

→ Checked Exception is an exception recognized by the Compiler but not occurred at Compilation time, Occurred at runtime.

Unchecked Exception is an exception recognized by JVM and Occurred at runtime.

Note:- In java applications, exceptions are always occurred at runtime Only, not at compilation time. While compilation of java programs, Compiler may predict some exceptional situation in our Coding part, but not occurred really at compilation time, this type of exceptions are called as checked exceptions.

Note:- In the above exception classes, Runtime exception and its sub classes, Error and its sub classes are come under Unchecked Exceptions and the remaining exception classes are come under checked exception.

→ There are two types of checked exceptions.

- (1) Pure Checked Exceptions
- (2) Partially Checked Exceptions.

Q: What is the diff. between pure checked exception and partially checked exception?

→ If any checked exception is having only checked exceptions as sub classes then that checked exception is called as pure Checked Exception.

Ex:- IOException.

30/10/2015

(82)

(33)

If any checked exception is having at least one sub class

of unchecked exception then that checked exception is called

Partially Checked Exception.

Ex:- Exception, Throwable.

## Overview of Predefined Exceptions:-

### ① ArithmeticException:-

In java applications, when we have a situation like a number divided by zero / num/0 where JVM will raise ArithmeticException.  
Ex:-

```
Class Test {
 public static void main(String[] args) {
 int a = 100;
 if b = 0;
 float f = a/b;
 System.out.println(f);
 }
}
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

If we execute the above program JVM will provide the following exception message.

Exception in thread "main" java.lang.ArithmaticException: / by zero  
at Test.main(Test.java:7)

The above exception is divided into the following three parts.

① Exception Name: java.lang.ArithmaticException

② Exception Description: / by zero

③ Exception Location: Test.java:7

## ② NullPointerException :-

In java applications, when we access instance variables or instance methods on a reference variable contains null value then JVM will rise an exception like NullPointerException.

Ex:-

```
Class Test{
 public void main(String[] args){
 java.util.Date d=null;
 System.out.println(d.toString());
 }
}
```

If we execute the above program then JVM will provide the following exception details.

① Exception Name: java.lang.NullPointerException

② Exception Description: -----

③ Exception Location: Test.java:6

## ④ ClassCastException :-

In java applications, we are able to keep sub class object reference value in super class reference variable, but we are unable to keep super class object reference value in sub class reference variable. If we are trying to keep super class reference value in sub class reference variable then JVM will rise ClassCastException.

Ex:-

```
Class A
{
}

Class B extends A
```

```
Class Test {
 public void main(String[] args){
 A a=new A();
 B b=(B)a;
 }
}
```

(33)

(34)

If we run the above program then JVM will provide the following exception details.

- ① Exception Name: java.lang.ClassCastException
- ② Exception Description: A cannot be cast to B
- ③ Exception Location : Test.java:12
- ④ ArrayIndexOutOfBoundsException:-

In java applications, when we access an element from a particular index value or when we insert an element at a particular index value, where if the specified index value is not in the range of the index value of an array then JVM will rise an exception like ArrayIndexOutOfBoundsException.

Ex:-

```
Class Test {
 public void main(String[] args) {
 int[] a = {1, 2, 3, 4, 5};
 System.out.println(a[10]);
 }
}
```

If we run the above program then JVM will provide the following exception details.

- ① Exception Name: java.lang.ArrayIndexOutOfBoundsException.
- ② Exception Description: 10
- ③ Exception Location : Test.java:6



## ⑤ ClassNotFoundException:-

In java applications, if we want to load a particular class bytecode to the memory then we have to use forName() method from java.lang.Class.

Ex:-

```
Class c=Class.forName("A");
```

When JVM encounter the above problem (instruction), JVM will search for the specified class .class file at current location at java predefined library and at the locations referred by "Classpath" environment variable. If the required Class file is not available at all these locations then JVM will rise an exception like "ClassNotFoundException".

Ex:-

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

```
 Class c=Class.forName("A");
}
```

If we run the above program then JVM will provide the following messages.

- ① Exception Name: java.lang.ClassNotFoundException
- ② Exception Description: A
- ③ Exception Location: Test.java:5



## ⑥ InstantiationException:-

(84)

(25)

In java applications, after loading a particular bytecode to the memory by using `Class.forName()` method if we want to create an object for the loaded class then we have to use the following method.

```
Class c = Class.forName("A");
```

```
Object obj = c.newInstance();
```

When JVM encounter `c.newInstance()` method then JVM will search for 0-arg constructor in the loaded class, if it is not available then JVM will rise `InstantiationException`.

Ex:-

```
Class A
```

```
{
```

```
 A (int i)
```

```
}
```

```
{
```

```
}
```

```
}
```

```
Class Test {
```

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

```
 Class c = Class.forName("A");
```

```
 Object obj = c.newInstance();
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

Doubt → As we know that JVM will create a 0-arg constructor for the class, where constructor is not defined, then how can it be possible, if constructor is not available in loaded then JVM will rise `InstantiationException`.

Cleared

If we run the above program then JVM will provide the following exception details.

- ① Exception Name: `java.langInstantiationException`
- ② Exception Description: A
- ③ Exception Location: Test.java: 12 //

here, it is find for 0-arg constructor but in class A, parameterized constructor is defined, so, JVM can't create the 0-arg constructor for class A.

31-10-2015

## ⑦ IllegalAccessException:-

In java applications, if we load class bytecode to the memory by using Class.forName() method and if we want to create object for the loaded class we have to use newInstance() method of java.lang.Class.

Ex:-

```
Class c = Class.forName("A");
```

```
Object obj = c.newInstance();
```

When we execute c.newInstance() method, JVM will gets the loaded class and JVM will search for non-private constructor. If the constructor is private constructor then JVM will raise exception like IllegalAccessException.

Ex:-

```
Class A {
```

```
 private A()
```

```
{
```

```
}
```

```
}
```

```
Class Test {
```

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

```
 Class c = Class.forName("A");
```

```
 Object obj = c.newInstance();
```

```
}
```

```
}
```

If we run the above program then JVM will provide the following exception details.

Exception Name: java.lang.IllegalAccessException.

Exception Description: Class Test cannot access a member of class A with modifiers "private".

Exception Location: Test.java:12

## ⑧ IllegalArgumentException:-

(85)  
(30)

In java applications, all the threads are having their own priority value, but the range of the priority values are 1 to 10. To set a particular priority value to the thread we have to use setPriority(value) method, where value must be provided within the range from 1 to 10 values. If we provide any value in outside range of 1 to 10 then JVM will raise IllegalArgumentException.

Ex:-

```
Class Test{
 public void main(String [] args){
 Thread t=new Thread();
 t.setPriority(15);
 System.out.println(t.getPriority());
 }
}
```

If we run the above program then JVM will provide the following exception message.

Exception Name:.. java.lang.IllegalArgumentException

Exception Description: - - - - -

Exception Location: Test.java:7



## throw keyword:-

In java applications, exceptions are raised without having developer's intention. In java applications, if we want to raise the exception intentionally as per the application requirement then we have to use "throw" keyword.

### Syntax:-

```
throw new Exception-Name ([Param-List]);
```

where Exception-Name may be predefined exception class or User defined exception class.

≡

### Ex:-

```
import java.io.*;
class Test {
 public void main (String args) throws Exception {
 BufferedReader br = new BufferedReader (new InputStreamReader
 (System.in));
 System.out ("Account number:");
 String aeno = br.readLine();
 System.out ("Account name:");
 String accName = br.readLine();
 System.out ("Enter pin number:");
 int pno = Integer.parseInt (br.readLine());
 System.out ("Account details");
 System.out ("-----");
 System.out ("Account Number: " + aeno);
 System.out ("Account Name : " + accName);
 System.out ("PIN Number : " + pno);
```

```
if (PIN < 1000 || PIN > 9999)
```

```
}
```

```
throw new RuntimeException ("Invalid PIN number, Please
enter valid PIN number");
```

```
}
```

```
else
```

```
{
 System.out.println ("Valid PIN number");
}
```

Note:- If we provide any statement immediately after "throw" statement then Compiler will rise an error like "Unreachable Statement", because it may never be reached to that statement.

### ① throws keyword:-

The main purpose of "throws" keyword is to bypass exception from present method to caller method in order to handle.

- In java applications, we have to use "throws" keyword in the methods prototype and Constructors prototype. → what is method prototype & constructor
- "throws" keyword are able to allow more than one prototype exception class name in method/Constructor syntax. (doubt)

### Syntax:-

```
Void m1 () throws ArithmeticException, ClassCastException,
NullPointerException
```

```
{
```

```
}
```

```
2
```

Eg:-

```
import java.io.*;
class A{
 void add() throws Exception
 {
 concat();
 }
 void concat() throws Exception
 {
 throw new IOException("my IOException");
 }
}

class Test{
 public main(String[] args) throws Throwable
 {
 A a=new A();
 a.add();
 }
}
```

If we compile the above program then Compiler will provide IOException notification inside concat() method, due to "throws" keyword in concat() method the generated exception is bypassed to add() method due to throws keyword in add() method prototype exception will be bypassed to main() method, due to throws keyword in main() method prototype exception is bypassed to caller of main() method that is JVM.

If we execute the above application then JVM will display exception message at all five location.

>java Test

Exception in thread "main" java.io.IOException; My IOException

at A.concat(Test.java:10)

at A.add(Test.java:6)

at Test.main(Test.java:18)

Q. What are the diff. between throw and throws keyword?

→ ① throw keyword can be used to raise an exception intentionally.

throws keyword can be used to express exception from present method to caller method in order to handle, where at caller method its rethrown exception must be handled either by using throws keyword again or by using try-catch-finally.

② throw keyword is used in method body, not in method declarative part or prototype.

throws keyword is used in method prototype or in method declarative part, not in method body.

③ throws keyword is able to allow only one exception class name. throws keyword is able to allow more than one exception class name.

### try-catch-finally:-

In java applications, 'throws' keyword is not really an exception handler, it will express the exception from present method to caller method, it will not handle exceptions really.

In java applications, if we want to handle the exception the locations where exceptions are generated then we have to use "try-catch-finally".

### Syntax:-

```
try
{
 — instructions —
}
```

Catch (Exception - Name e)

```
{
 — instructions —
}
```

finally  
{  
 — instructions —  
}

Where the main purpose of try block is to include some instructions [risky code] where there may be a chance to get an exceptions.

If we write some instructions in try block then it may not give any guarantee to generate exceptions, may rise or may not rise the exceptions.

If we get an exception in try block then JVM will bypass flow of execution to catch block with the generated exception object as parameter by skipping the remaining instructions in try block.

If no exception is generated by try block then JVM will forward (skip) flow of execution to finally block directly without executing catch block.

Note:- In "try" block JVM is able to generate one exception maximum, we are unable to get more than one exception from try block.

→ The main purpose of 'catch' block is to display exception details to see use on the command prompt. How can it possible?

→ To display exception details on command prompt we have to use the following three approaches. If catch block display

① e.printStackTrace();

the exception details like who is handle it and what is reason of these three modes.

→ It will display the exception details like name of the exception, description of the exception and location of the exception.

public void printStackTrace()

② System.out.println(e);

When we pass exception object reference variable as parameter to System.out.println() method then JVM will execute toString() method internally. Exception is provided its own toString() method, it was implemented in such a way that to return a String containing the exception details like name of the exception and description of the exception.

public String toString()

③ System.out.println(e.getMessage());

Where getMessage() method was implemented in such a way that to return a string excluding the exception details like name of the description of the exception.

public String getMessage()

Ex:-

Z

```

class Test
{
 public static void main(String[] args)
 {
 try
 {
 int i=100;
 int j=0; → if the value is 10^{-15}

 float f=i/j;
 }
 catch (Exception e)
 {
 e.printStackTrace();
 System.out.println(e);
 System.out.println(e.getMessage());
 }
 finally
 {
 }
 }
}

```

OP:- java.lang.ArithmeicException: / by zero  
           at Test.main(Test.java:9)  
           java.lang.ArithmeicException: / by zero  
           / by zero.

### \*finally:-

→ Here the main purpose of "finally" block is to include a set of instructions which must be executed by JVM irrespective of getting exception in try block and irrespective of executing catch block.

(40)

(87)

Note:- If we use resources like files, streams, database connections, ..... in java applications then it is mandatory to close these resources before terminating java apps, if we want to close these resources then we have to keep the corresponding resources Close() methods inside finally block, where finally block will give guarantee to execute Close() methods before terminating java applications.

Q. What is the differences between final, finally and finalize?

→ final is a java keyword, it can be used to declare constant expressions.

final Variables: Not to allow modifications on their values.

Final Methods: Not to allow overriding on their functionalities.

final Classes: Not to allow extensions [Inheritance] from the classes.

"finally" is block in "try-catch-finally" syntax, it will include a set of instructions to execute irrespective of getting exception in try block and irrespective of executing catch block.

"finalize()" is a method from java.lang.Object class, it will be executed by JVM just before destroying an Object in Garbage Collection in order to give final intention about to destroy object.

≡

04/10/2015

- In java apps, it is possible to provide "try" block without catch block but with finally block.

```
try
{
}
finally
{
}
```

- If we have exception in try block in the above example then JVM will take that exception and JVM will display the required exception details with the help of default exception handler internally.

- In java applications, it is possible to provide try block with out finally block but with catch block.

```
try
{
}
catch(Exception name e)
{
}
}
```

Q: Find the Output from the following program?

```
class Test {
 public static void main (String [] args){
 System.out.println ("Before try");
 try
 {
 System.out.println ("Inside try, before exception");
 }
```

90  
 91

```

 i=100;
 if j=0;
 float f = e/j;
 S.O.P1N(f)
 S.O.P1N("Inside try, after exception");
 {
 Catch(Exception e)
 }
 S.O.P1N("Inside catch");
 {
 finally
 }
 S.O.P1N("Inside finally");
 S.O.P1N("After finally");

```

O/P:- Before try  
 Inside try, before exception  
 Inside catch  
 Inside finally  
 After finally.

Q: find the Output for the following program?

```

class A {
 int m1()
 {
 try
 {
 return 10;
 }
 catch(Exception e)
 {
 return 20;
 }
 finally
 {
 return 30;
 }
 }
}

```

```

class Test {
 public static void main(String[] args)
 {
 A a = new A();
 int val = a.m1();
 S.O.P1N(val);
 }
}

```

O/P:- 30

In java apps, we are able to provide try-catch-finally inside try, inside catch and inside finally.

SYNTAX①:-

```
try
{
 try {
 catch (Exception e) {
 finally {
 }
 catch (Exception e)
 {
 }
 finally
 {
 }
 }
 }
 }
```

SYNTAX②:-

```
try
{
}
catch (Exception e)
{
 try {
 catch (Exception e)
 {
 finally {
 }
 }
 }
 }
}
```

SYNTAX③

```
try
{
}
catch (Exception e)
{
}
finally
{
 try {
 catch (Exception e) {
 finally {
 }
 }
 }
 }
}
```

91

92

In java applications, we can provide more than one catch block for a single try block with the following conditions.

- ① If inheritance relationship is existed between exception classes which are specified in catch blocks then we have to arrange all the catch blocks as per the exception classes inheritance increasing order.
- ② If inheritance relationship is not existed between exception classes which we specified along with catch blocks then we can arrange all the catch blocks in any order.
- ③ If any pure checked exception like IOException is specified along with any catch blocks then the corresponding try block must rise the same pure checked exception.

Ex: ①

```
try {
 } catch (ArithmaticException e) {
 } catch (NullPointerException e) {
 } catch (ClassCastException e) {
}
```

Status :- Valid

Ex: ②

```
try {
 } catch (NullPointerException e) {
 } catch (ClassCastException e) {
 } catch (ArithmaticException e) {
}
```

Status: Valid

Ex: ③

```
try {
 } catch (ArithmaticException e) {
 } catch (Runtimeexception e) {
 } catch (Exception e) {
}
```

Status:- Valid

≡

Ea:- ④

```
try {
} catch(Exception e){
} catch(RuntimeException e){
} catch(ArithmaticException e){
}
```

Status: Invalid. ≡

Ea:- ⑤

```
try {
 throw new ArithmaticException();
} catch(ArithmaticException e){
} catch(NullPointerException e){
} catch(IOException e){
}
```

Status: Invalid. ≡

Ea:- ⑥

```
try {
 catch
 throw new IOException();
} catch(ArithmaticException e){
} catch(NullPointerException e){
} catch(IOException e){
}
```

Status: Valid. ≡



## \* ② Custom Exception/User-defined Exception:-

These exceptions are defined by the developers as per their requirement.

To implement user defined exceptions in java applications we have to use the following steps.

### ① Prepare user defined exception class:-

a) declare an user defined class.

b) Extend java.lang.Exception class to User defined class in order to get exceptions' behaviour to user defined class.

c) In User defined exception class, declare String parameterized constructor in order to get Exception description at the time of creating Exception object.

d) In user defined exception class constructor, access super class [Exception] class String parameterized constructor by using "Super" keyword to set exception description to printStackTrace() method, toString() method and getMessage() method in order to display exception details on command prompt from catch block.

Ex:-

```
class InvalidMarksException extends Exception
{
 InvalidMarksException(String err-msg)
 {
 super(err-msg);
 }
}
```

## ② Rise User Defined exception and Handle User defined Exception:-

To rise user defined exception we have to use "throw" keyword. To handle user defined exception we have to use "try-catch-finally".

Ex:-

```
try
{
 if(smarks<0 || smarks>100)
 {
 throw new InvalidMarksException("Marks are invalid, please
 provide marks in the range from 0 to 100");
 }
}
catch(InvalidMarksException e)
{
 System.out.println(e.getMessage());
}
```

Ex:- ②

Class InsufficientFundsException extends Exception

```
{
 InsufficientFundsException(String err-msg)
 {
 super(err-msg);
 }
}
```

```
Class Transaction
{
 String accno;
```

Q1  
44

```

String accName;
String accType;
Transaction(String accNO, String accName, String accType, int
balance)
{
 this.accNO = accNO;
 this.accName = accName;
 this.accType = accType;
 this.balance = balance;
}
public void withdraw(int wd_Amt)
{
try
{
S.O.PIN("Transaction Details");
S.O.PIN("-----");
S.O.PIN("Transaction ID: T-111");
S.O.PIN("Account Number: " + accNO);
S.O.PIN("Account Name: " + accName);
S.O.PIN("Account Type: " + accType);
S.O.PIN("Transaction Type: WITHDRAW");
S.O.PIN("Initial Balance: " + balance);
S.O.PIN("Withdraw Amount: " + wd_Amt);
if (wd_Amt < balance)
{
 balance = balance - wd_Amt;
 S.O.PIN("Total Balance: " + balance);
 S.O.PIN("Transaction Status: SUCCESS");
} else
 S.O.PIN("Total Balance: " + balance);
}

```

```
s.o.pin("Transaction Status: FAILURE");
throw new InsufficientFundsException("Funds are not
sufficient in our account, please provide valid amount");
```

```
} } catch(InsufficientFundsException e)
{ s.o.pin(e.getMessage()); }
```

```
finally
```

```
{ s.o.pin(" * * * * ThanQ, visit Again * * * * "); }
```

```
} class Test {
```

```
PSVM (String[] args)
```

```
Transaction tx1 = new Transaction ("abc123", "Durga",
"savings", 2000);
```

```
tx1.withdraw(5000);
```

```
s.o.pin();
```

```
Transaction tx2 = new Transaction ("xyz123", "Mangoor",
"savings", 10000);
```

```
tx2.withdraw(15000);
```

```
} //
```

## JAVA7 Features in Exception Handling:-

(94)  
(95)

- ① Multi catch block.
- ② try-with-resources/Auto-Closeable Resources.

### ① Multi Catch Block :-

In java applications, to handle exceptions, we will use "try-catch-finally", where if we use `java.lang.Exception` class in catch block then it able to handle all the exceptions which are same as `java.lang.Exception` or which are sub classes to `java.lang.Exception`.

```
try {
 ...
}
catch(Exception e){
 ...
}
```

In the above approach, catch block is having all the exceptions commonly, it is not providing individual procedure to handle exception.

In java applications, if we want to handle the exceptions individually then we have to use multiple catch blocks for a single try block.

```
try {
 ...
}
catch(Exception1 e){
 ...
}
catch(Exception2 e){
 ...
}
catch(Exception3 e){
 ...
}
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

If we use the above approach then ~~we have~~ no. of Catch blocks are increased, Code length will be increased.

In java applications, if we want to handle all the exceptions, individually by using single catch block then we have to use "Multi-Catch Block" provided by JAVA7 Version.

### Syntax:-

```
try {

} catch(Exception1 | Exception2 | Exception-n | ref-var) {

}
```

Where, Exception1, Exception2, ... Exception-n must be individual exception classes, they must not have inheritance relation, if inheritance relation exists between all the exception classes then compiler will rise an error.

### Exit Class Test

```
PSVM(String [] args) {
 try {
 /*
 * exception handling
 */
 int i=100;
 int j=0;
 float f=i/j;
 }
 /*
 * java.util.Date d=null;
 */
 System.out.println(d.toString());
}
```

06/11/2015

(95)

(96)

```
int[] a={1,2,3,4,5};
System.out.println(a[10]);
```

```
}
catch(ArithmaticException | NullPointerException | ArrayIndexOutOfBoundsException
Exception e)
{
 e.printStackTrace();
}
```

Note:- This program will work in only JAVA7,  
JAVA8 not work in JAVA6.7.

06/11/2015

### try-with-resources:- / Auto-Closeable Resources:-

In general, In java apps, we are able to use the resources like files, Streams, database Connections,..... as per the requirement. In java apps, if we want to manage all the resources along with "try-Catch-Finally" then we have to use the following conventions.

- ① Declare resources before try block.
- ② Create the resources inside try block.
- ③ Close the resources inside finally block.

Ex:-

```
file f=null;
BufferedReader br=null;
Connection con=null;
try{
 f=new File("abc.txt");
 br=new BufferedReader(new InputStreamReader(System.in));
 con=DriverManager.getConnection(...,...,...);
 --
```

```
catch (Exception e) {
```

```
 --
```

```
 }
```

```
 finally {
```

```
 try {
```

```
 f.close();
```

```
 br.close();
```

```
 con.close();
```

```
} catch (Exception e) {
```

```
 --
```

```
 }
```

```
}
```

In the above convention, to close the resources developers have to use close() methods explicitly, where close() methods may throw out some exceptions like IOException or SQLException, to handle these exceptions again we have to use try-catch - finally inside finally and developers may remember or may not remember to write close() methods explicitly.

To overcome all the above problems, JAVA has provided an alternative in the form of "try-with-resources" or "Auto-Closeable-Resources".

In case of try-with-resources, we have to declare and create all the resources along with "try", not before try block and not inside try block, here, if we declare the resources along with try then try will close all these resources automatically when ever JVM is coming out from try block, in this case, it's not required to close the resources explicitly by using close() methods.

If we want to declare and create the resources along with "try" then that resources must implement or extend `java.io.AutoCloseable` marker interface.

Note:- In JAVA7, almost all the Stream classes, Connection interface, ... are implemented `java.io.AutoCloseable` marker interface.

If we declare the resources along with try then the resources reference variables are converted as "final" variables.

SYNTAX:-

```
try(Resource1;Resource2;Resource-n)
```

{

---

}

---

for:

```
try(File f = new File("abc.txt"));
```

```
BufferedReader br = new BufferedReader(new InputStreamReader
(System.in));
```

Connection Con = DriverManager.getConnection(--, --, --);

)

---

}

```
Catch(Exception e){}
```

---

}

==

Ex:-

```
import java.io.*;
public class TryWithResourceex {
 public static void main(String[] args) throws Exception {
 try (FileInputStream fis = new FileInputStream("abc.txt");
 FileOutputStream fos = new FileOutputStream("xyz.txt")) {
 int size = fis.available();
 byte[] b = new byte[size];
 fis.read(b);
 fos.write(b);
 System.out.println("Data is transferred from abc.txt file to xyz.txt file");
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
}
```

Exception handling finished.

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

06/11/2015

(47)

(48)

## Inner Classes:-

Declaring a class inside another class is called as Inner class.

Ex:-

Class Outer{

    Class Inner{

    -----

}

In java applications, inner classes are able to provide the following advantages.

### ① Modularity:-

If we want to define modules over the application logic with in a single class then we can use Inner classes.

Ex:-

Class Account{

    Class StudentAccount{

    -----

}

    Class EmployeeAccount{

    -----

    Class LoanAccount{

    -----

}

    }

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## ② Abstraction:-

If we declare variables and methods in an inner class they that variables and methods are available upto the respective inner class, they are not available to other inner class, so that inner classes are able to improve abstraction.

## ③ Security:-

In java applications, we are unable to declare a class as private class but we are able to declare an inner class as private inner class, so that inner classes are able to improve security.

## ④ Sharability:-

In java applications, we are unable to declare a class as static, but, we are able to declare an inner class as static, so that inner classes are able to improve sharability.

## ⑤ Reusability:-

In java apps, "Inheritance" relation between classes is able to improve Reusability. In java apps, it is possible to extend one inner class to another inner class, so that inner classes are able to improve Reusability.

If we compile java file then compiler will create a separate .class file for Outer class and a separate .class file for Inner class.

Outer class .Class file : Outer.class

Inner class .Class file : Outer\$Inner.class

98 49

There are four types of inner classes in java applications.

- ① Member Inner Class
- ② Static Inner Class
- ③ Method Local Inner Class
- ④ Anonymous Inner Class.

### ① Member Inner class:-

Declaring a normal (non-static) class in another class is called  
as Member Inner Class.

Syntax:-

Class Outer{

    Class Inner{

        }

If we want to access data from member inner classes, we have to use the following instruction.

Outer.Innerobj = new Outer().new Inner();

If we declare data in outer class then we are able to use that data directly inside the inner class. If we declare data in member inner class then we are unable to use that data in outer class.

In java apps, by using outer class reference variable we are able to access only outer class members, we are unable to access inner class members. By using inner class reference variable we are able to access only inner class members and we are unable to access outer class members.

~

07/11/2015

In case of member inner classes, static declarations are not allowed directly, but, static keyword is allowed along with final keyword.

In ~~java applications~~, for

Ex:-

```
Class A {
 int i=30;
 void m1()
 {
 System.out.println("m1-A");
 // System.out.print(j); → Error
 }

Class B {
 int j=20;
 // static int k=30; → Error
 static final int k=30;
 void m2()
 {
 System.out.println("m2-B");
 System.out.print(i);
 System.out.print(k);
 }
 void m3()
 {
 System.out.println("m3-B");
 }
}

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

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

```

A a=new A();
a.m1();
//a.m2(); → Error
//a.m3(); → Error

```

~~A.B ab = new A();~~

```

A.B ab = new A().new B();
//ab.m1();
ab.m2();
ab.m3();

```

{  
}

=

→ In java applications, it's possible to extend one inner class to another inner class but both inner classes must be available with in the same outer class.

Ex:-

```

class A {
 class B {
 void m1() {
 System.out.println("m1-B");
 }
 }
}

class C extends B {
 void m2() {
 System.out.println("m2-C");
 }
}

```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

```

class Test {
 public static void main(String[] args) {
 A.B ab = new A().new B();
 ab.m1();
 A.C ac = new A().new C();
 ac.m1();
 ac.m2();
 }
}

```

Ex:-

Class A {

Class B {

void m1() {

S.o.p1n("m1-B");

}

Class D extends A,B {

void m2() {

S.o.p1n("m2-D");

O/P:- CE

Ex:-

Class A {

Void m1() {

S.o.p1n("m1-A");

}

Class C extends A {

Void m2() {

S.o.p1n("m2-C");

}

PSVM(String[] args) {

A a = new B().new C();

a.m1();

B.C bc = new B().new C();

bc.m1();

bc.m2();

O/P:- Valid. → Status.

Ex:-

Class A {

Class B extends A {

-----

}

}

Status : valid.

Ex:-

Q: Is it possible an interface inside a class?

100

51

⇒ Yes, it is possible to provide an interface inside the class, but the respective implementation class must be provided with in the same outer class.

Ex:-

```
class A {
 Interface I {
 void m1();
 void m2();
 }

 class B implements I {
 public void m1() {
 System.out.println("m1-B");
 }

 public void m2() {
 System.out.println("m2-B");
 }
 }

 Class Test {
 public static void main(String[] args) {
 A.I ob = new A().new B();
 ob.m1();
 ob.m2();
 }
 }
}
```

## Static Inner Classes:-

Declaring a static class inside a class is called as static inner class.

### Syntax:-

```
Class Outer {
 static class Inner
```

```
{

}
```

If we want to access the members from static inner classes then we have to create object for static inner class, for this, we have to use the following syntax.

```
Outer.Inner ref_var = new Outer.Inner();
```

In java applications, static inner classes are able to allow only static members of the outer classes, not non-static members of the outer class.

In general, inner classes are not allowing static declaration, but, static inner classes are allowing static declarations directly.

```
Ex:- Class A {
 int i=10;
 static int j=20;
 static class B
 {
```

(10)  
(52)

```
Void m1()
{
 S.O.P("m1-B");
 //S.O.P("x"); → Error
 S.O.P("m1(j)");
}

Void m2()
{
 S.O.P("m2-B");
}

Static Void m3()
{
 S.O.P("m3-B");
}
```

```
Class Test{
 Psvm(String[] args){
 A.B ab = new A.B();
 ab.m1();
 ab.m2();
 A.B.m3();
 }
}
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Q Is it possible to provide a class inside an interface?

⇒ Yes, it is possible to provide a class inside an interface.

If we declare a class inside an interface then that class is by default static either class inside the interface. We can access the members of that inner class like static inner class.

Ex:-

```
interface I {
 class A {
 void m1()
 {
 System.out.println("m1-A");
 }
 void m2()
 {
 System.out.println("m2-A");
 }
 }
 class Test {
 public static void main(String[] args) {
 I.A ia = new I.A();
 ia.m1();
 ia.m2();
 }
 }
}
```

### ③ Method local Inner Class:-

102  
53

Declaring a class inside a method is called as Method local Inner Class. If we declare an inner class inside a method then that inner class is having scope upto the respective method like local variable.

If we want to access the members of method local inner class then we have to create object for method local inner class inside the respective method only.

Eg:-

```
class A {
```

```
 void m1()
```

```
{
```

```
 class B
```

```
{
```

```
 void m2()
```

```
{
```

```
 System.out.println("m2-B");
```

```
}
```

```
 void m3()
```

```
{
```

```
 System.out.println("m3-B");
```

```
}
```

```
 B b = new B();
```

```
 b.m2();
```

```
 b.m3();
```

```
} // m1()
```

```
// Class A
```

Class Test {

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

```
 A a = new A();
```

```
 a.m1();
```

```
}
```

08/11/2015

## Anonymous Inner Class:-

Anonymous inner classes are nameless inner classes, which can be used to provide implementations for interfaces and abstract classes.

In java apps, for interfaces we are able to take implementation classes, but implementation classes are able to allow their own members and implementation class objects are having their own identity.

In java apps, if we want to have an alternative to allow only interface members implementation and to have only interface identity we have to use ~~any~~ anonymous inner class.

### Syntax:-

```
interface name/abstract class Name
{

}
Class Outer
{
 Name ref-var = new Name()
 {
 --- Implementation for interface/abstract class members
 };
}
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery  
Opp: CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

103

59

Ex:- (Anonymous inner classes with interface).

interface I  
{  
    void m1();  
    void m2();  
    void m3();  
}

Class A {  
    I i = new I()  
    {  
        public void m1()  
        {  
            System.out.println("m1-AIC");  
        }  
        public void m2()  
        {  
            System.out.println("m2-AIC");  
        }  
        public void m3()  
        {  
            System.out.println("m3-AIC");  
        }  
        public void m4()  
        {  
            System.out.println("m4-AIC");  
        }  
    };  
}

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Class Test {  
    public static void main(String[] args){  
        A a = new A();  
        a.i.m1();  
        a.i.m2();  
        a.i.m3();  
        // a.i.m4(); → error  
    }  
}

Ex:- (Anonymous Inner Classes with Abstract Class)

Abstract class A {

    void m1()

    {  
        S.O.P1("m1-A");  
    }

    abstract void m2();

    abstract void m3();

}

Class Outer {

    A a = new A()

    {  
        void m2()

        {  
            S.O.P1("m2-AIC");  
        }

        void m3()

        {  
            S.O.P1("m3-AIC");  
        }

    };

}

Class Test {

    PSVM(String[] args) {

        A a = new A();

        Outer o = new Outer();

        o.a.m1();

        o.a.m2();

        o.a.m3();

→ In java applications, if we declare an interface with an abstract method and if we want to pass interface reference as parameter to some methods, then we have to pass Anonymous inner class object reference directly as parameters instead of taking implementation class and instead of passing implementation class object reference value.

Ex:-

```
interface I {
 void m1();
}
```

```
Class A {
 void m2(I i) {
 System.out.println("m2-A");
 i.m1();
 }
}
```

```
Class Test {
 public static void main(String[] args) {
 A a = new A();
 a.m2(new I() {
 public void m1() {
 System.out.println("m1-A IC");
 }
 });
 }
}
```

→ Anonymous inner class  
Reference



## Multi Threading:-

Q What is the difference between Process, Procedure and Processor?

→ Process is a flow of executing to execute a set of instructions in order to perform a particular task.

Procedure is a set of instructions to represent a particular task.

Processor is an HW component to generate no. of processes in order to execute no. of tasks.

As the starting point of the Computers, we have single process mechanism or Single tasking to execute applications.

In case of single process mechanism, system will allow only one task to load in memory even though memory is capable to load all the tasks and System will create a single process to execute all the tasks in One by One fashion that is in sequential execution. This approach is not using system memory effectively and it will take more execution time to execute applications and it will reduce application performance.

To overcome all the above problems we have to use Multi Processing System or Multi Tasking. In case of Multi Tasking, System will load all the tasks to the memory at a time, it will allow to create a separate process for each and every task and it will start all the process to execute the respective time at a time, this approach is following parallel execution, it will reduce execution time and it will improve application performance.

To perform Multi Tasking we need the following Components.

① Memory:- To store all the tasks in order to execute.

## ② Process Waiting Queue:-

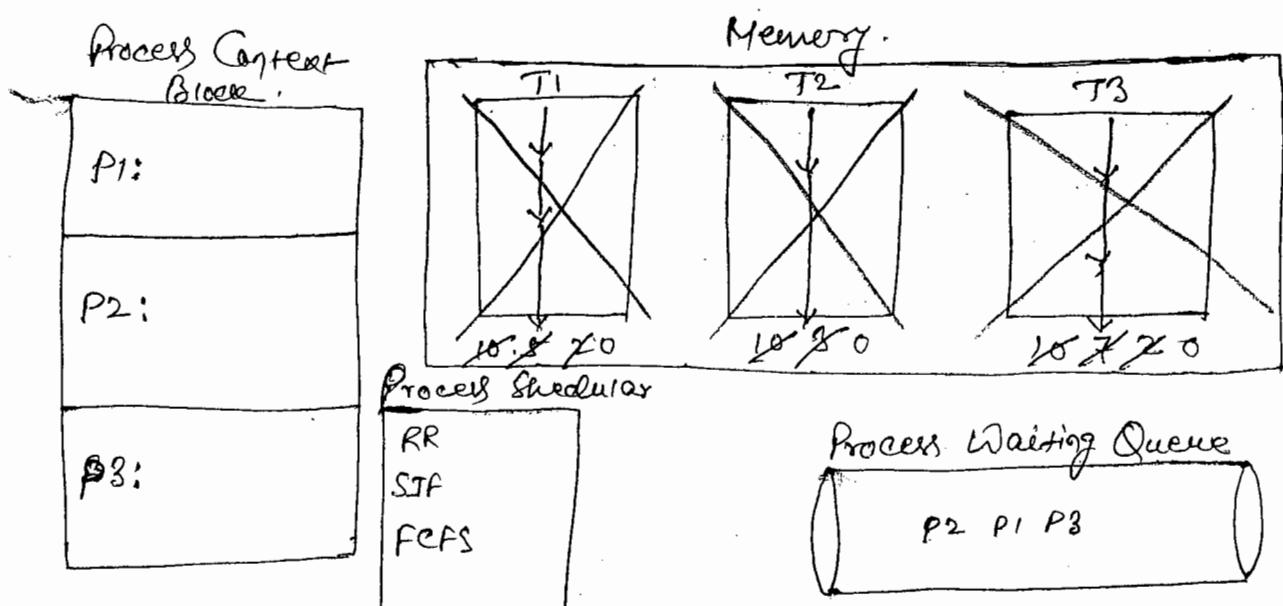
To keep trace of all the processes to execute tasks.

## ③ Process Context Block:-

To manage the status of each and every process which are executing.

## ④ Process Scheduler:-

It will take processes from process waiting queue and it will assign time stamps for each and every process in order to execute tasks.



In Multi tasking controlling is switched from one process context to another process context, it's called as Context Switching.

There are two types of context switching:

- ① Heavy Weight Context Switching.
- ② Light Weight Context Switching.

## ① Heavy Weight Context Switching:-

(59)

It is the Context switching between two heavy weight Components, it will take more execution time and it will reduce application performance.

Ex:- Context Switching between processes.

## ② Light Weight Context Switching:-

It is the Context switching between two light weight Components, it will take less execution time and it will improve application performance.

Ex:- Context switching between two threads.

\* Q: What is the diff. between Process and Thread?

→ Process is heavy weight, it will consume more system memory and more execution time and it will reduce appd performance.

Thread is light weight, it will take less s/m memory and less execution time and it will improve application performance.

→ There are two thread Models.

### ① Single Thread Model:-

It able to allow only one thread at a time to execute applications, it will take more execution time and it will reduce application performance.

### ② Multi Thread Model:-

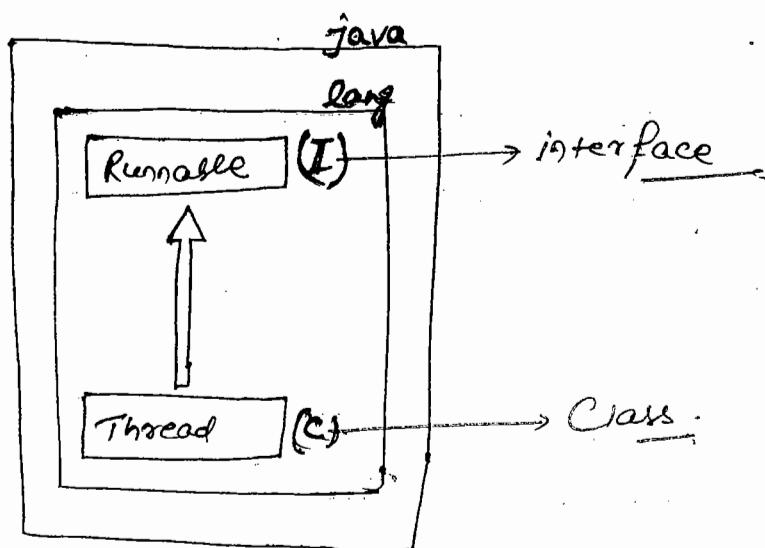
It able to allow more than one thread at a time to execute more than one task, it will take

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

less execution time to execute apps, and it will increase performance of the apps.

Java is following Multi Thread Model to execute applications, it able to allow to create and execute more than one thread at a time.

To create Threads, JAVA has provided the following predefined library in java.lang package.



Q: What is thread and in how many ways we are able to create threads in java applications?

→ Thread is a flow of execution to perform a particular task.

As per few predefined library provided by JAVA, there are two ways to prepare threads.

① Extending Thread Class:-

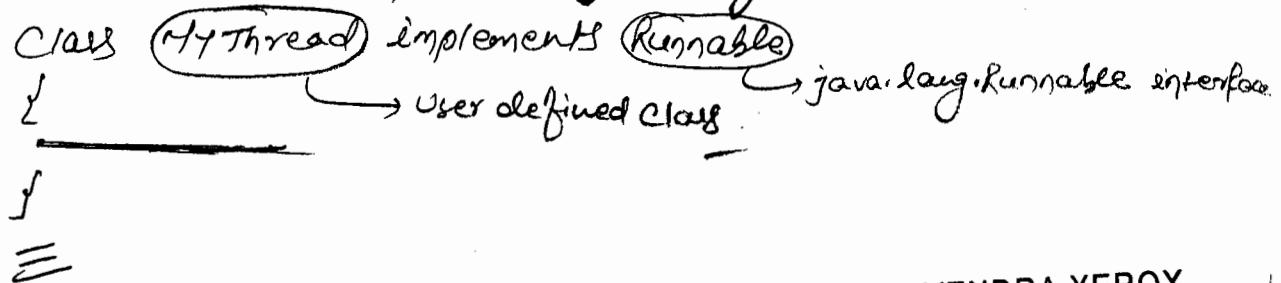
In this approach, we have to declare an user defined class, it must be extended from java.lang.Thread class.

Class MyThread extends Thread  
User defined class

## ② Implementing Runnable Interface:-

(60)

In this approach, we have to declare an user defined class, it must implement java.lang.Runnable interface.



### Threads Design:-

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

- ① Extending Thread Class
- ② Implementing Runnable Interface.

### ① Extending Thread Class:-

- a) Declare an user defined class.
- b) Extend java.lang.Thread class to User defined class.
- c) Provide application logic which we want to execute by creating new thread in user defined thread class by overriding run() method.  
`public void run()`
- d) In main class, in main() method, Create Object for User defined thread class.
- e) Access start() method on user defined thread class object reference.

`public void Start()`

The main purpose of start() method is to create a new flow of execution [thread] and access user defined thread class run() method by bypassing newly created thread.

Z

Q:-

Class MyThread extends Thread

{  
    public void run()  
    {

        for (int i=0; i<10; i++)

            System.out.println("User Thread");

}

Class Test { → Thrd.java }

    public String[] args) {

        MyThread mt = new MyThread();

        mt.start();

        for (int i=0; i<10; i++)

            System.out.println("Main Thread");

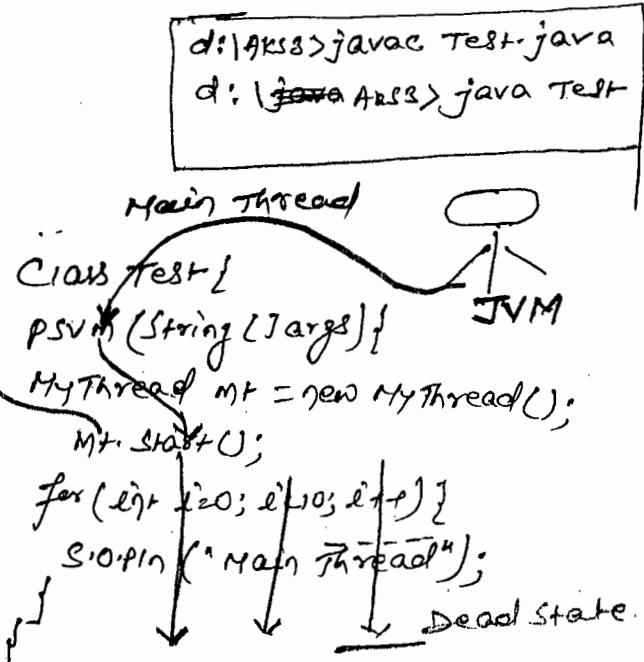
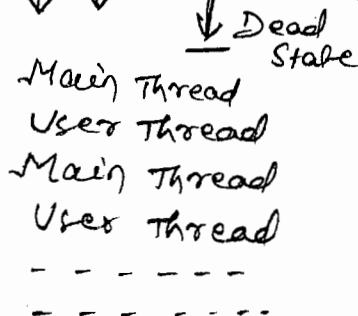
without start() method we can't  
access the user defined class  
without start() method only  
"Main Thread" will be executed

Z  
Z

```

class MyThread extends Thread {
 public void run() { User Thread
 for (int i=0; i<10; i++) {
 System.out.println("User Thread");
 }
 }
}

```



**Q.** In java applications, to create threads we have already first approach [extending Thread class] then what is the requirement to go for second approach [Implementing Runnable interface]?

→ In java apps, if we use first approach to create threads then we have to extend java.lang.Thread class to our defined class, in this case, if it is not possible to extend any other classes to the user defined class, if we extend any other class along with Thread class then it will represent multiple inheritance, it's not possible in java.

→ To Overcome the above problem we have to use second approach to create thread, in this case we can extend any other class and we can implement java.lang.Runnable interface so make\*

If we want to prepare thread in first approach then we have to extend Thread class to an user defined class.

Class MyClass extends Thread

```
class MyClass extends Thread
{
 //
}
```

If we want to prepare frame in GUI apps then we have to extend java.awt.Frame class to user defined class.

Class MyClass extends Frame

```
class MyClass extends Frame
{
 //
}
```

If we want to create Thread and frame by using a single class then we have to declare an user defined class, it must be extended from java.awt.Frame class and java.lang.Thread class.

Class MyClass extends Frame, Thread.

```
class MyClass extends Frame, Thread
{
 //
}
```

The above situation is representing Multiple Inheritance, it is not possible in java. In this context, to resolve multiple inheritance, we have to use Second approach to create thread that is implementing java.lang.Runnable interface.

Class MyClass extends Frame implements Runnable

```
class MyClass extends Frame implements Runnable
{
 //
}
```

## ② Implementing Runnable interface:-

- Declare an User defined class
- Implement java.lang.Runnable interface in user defined class.
- Provide application logic in user defined class by implementing run() method.
- In main class, in main() method create new thread and access run() method.

To achieve this, we have to use the following cases.

Class MyThread implements Runnable

```

 {
 public void run()
 }
}

```

### Case:- ①

MyThread mt = new MyThread();

mt.start();

Status:- CE

Reason:- Start() method is not defined in MyThread class,  
it is defined in java.lang.Thread class.

### Case:- ②

MyThread mt = new MyThread();

mt.run();

Status:- NO CE, but main thread is calling run()

method like a normal java method, it's got multi threading.

2

Note:- In java apps, we are unable to create threads without using Thread class start() method.

Case 3 :-

MyThread mt = new MyThread();

Thread t = new Thread(mt) → New thread is created

    t.start(); → Thread class start() method.

Status:- No CE, here new thread is created by start() method, but start() method is executing by thread class run() method, it is not executing MyThread class run() method.

Case: ④

MyThread mt = new MyThread();

Thread t = new Thread(mt);

    t.start();

Status:- No CE, here start() method will create a new thread, start() method will access user defined thread class run() method by passing newly created thread.

Ex:-

class MyThread implements Runnable

{  
    public void run()  
    {

        for (int i=0; i<10; i++)

            System.out.println("User Thread");

(63)

```
class Test {
 public static void main(String[] args) {
 MyThread mt = new MyThread();
 mt.start();
 }
 MyThread mt = new MyThread();
 mt.run();
}
MyThread mt = new MyThread();
Thread t = new Thread(
 mt);
t.start();
MyThread mt = new MyThread();
Thread t = new Thread(mt);
t.start();
for (int i=0; i<10; i++)
{
 System.out.println("Main Thread");
}
```

3 Copied... (flow of execution)

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

```

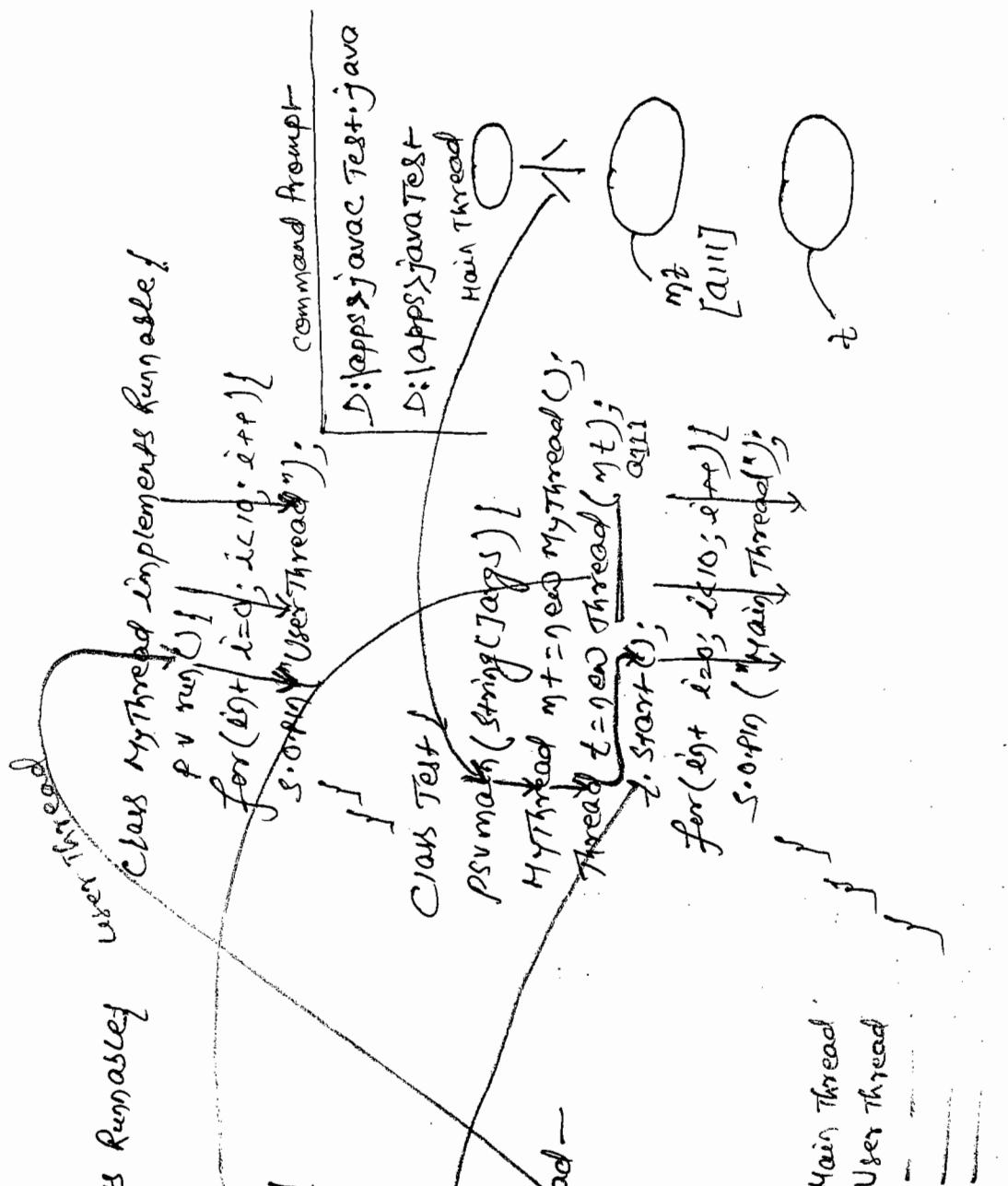
public class Thread implements Runnable {
 Runnable r = null;
 public Thread() {
 }
 public Thread(Runnable r) {
 this.r = r;
 }
 public void run() {
 public void start() {
 logic to create new thread -
 if (x == full) {
 this.run();
 } else {
 run();
 }
 }
 }
}

```

## FLOW OF Execution Of Above example.

reference value of `MyThread`

(14)



## Thread Lifecycle:-

The collective information of a thread right from its starting point to ending point is called as Thread Lifecycle.

In java applications, Threads are available in the following five states.

### ① New/Born State:-

When we create a Thread class object, automatically, Thread will come to "New/Born" state.

### ② Ready/Runnable State:-

When we access start() method over the Thread reference variable and CPU is assigning system resources to the thread then the state of the thread is "Ready/Runnable" state.

### ③ Running State:-

After calling start() method and when CPU is allocating system resources then the state of the thread is "Running" state.

### ④ Blocked State:-

In java apps, we are able to keep a running into blocked state, in the following situations.

- ① When we access ~~start()~~ Sleep() method.
- ② When we access wait() method
- ③ When we access suspend() method.
- ④ When we perform IO Operations.

In java apps, we are able to get a thread from Blocked state to Runnable state in the following situations.

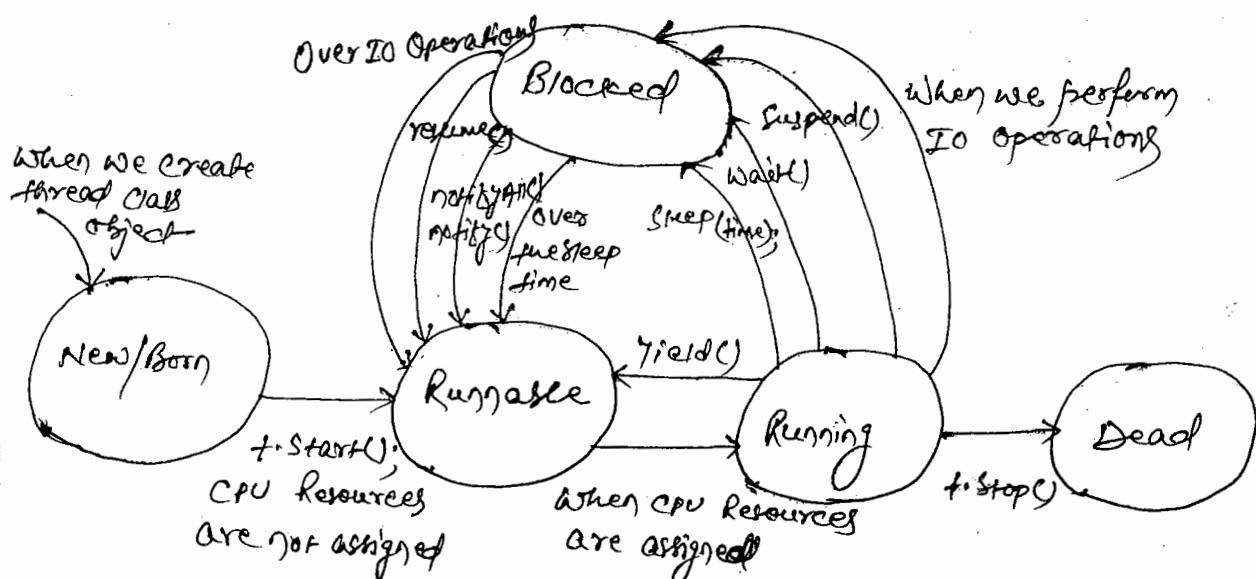
- ① When Sleep time is over.
- ② When notify() or notifyAll() methods are accessed by other threads.

③ When resume() method is accessed by other threads.

④ When IO Operations are completed.

⑤ Dead State:-

When we access Stop() method over Running thread then that



## Thread Class Library:-

### Constructors:-

#### ① Public Thread()

→ This constructor can be used to create Thread class object with the following default properties.

Thread Name: Thread-0

Thread Property: 5

Thread Group: main

Eg:-

Thread t = new Thread();

S.O.P(t);

Op:- Thread[Thread-0, 5, main] //

## ② Public Thread(Runnable r)

→ This constructor can be used to create Thread class object with the specified Runnable reference.

Ex:-

Runnable r = new Thread();

Thread t = new Thread(r);

S. O. P() t;

O/P:- Thread[Thread-1, 5, main]

## ③ public Thread(String name)

→ It can be used to create Thread class object with the Specified name.

Ex:-

Thread t = new Thread("Core Java");

S. O. P() t;

O/P:- Thread[Core Java, 5, main]

## ④ public Thread(Runnable r, String name)

→ It can be used to create Thread class object with the Specified Runnable reference and Specified thread name.

Ex:-

Runnable r = new Thread();

Thread t = new Thread(r, "Core Java");

S. O. P() t;

O/P:- Thread[Core Java, 5, main]

## ⑤ public Thread(ThreadGroup tg, Runnable r)

→ This constructor can be used to create Thread class object with the Specified ThreadGroup name and Runnable Reference.

In java applications, to represent Thread Group Name, JAVA has provided a predefined class in the form of "java.lang.ThreadGroup". To create ThreadGroup class Object, we have to use the following Constructors.

public ThreadGroup(String name)

Ex:-

Runnable r = new Thread();

ThreadGroup tg = new ThreadGroup("Java");

Thread t = new Thread(tg, r);

S.O.P() (t)

O/P:- Thread[Thread-1, 5, main].

(6) public Thread(ThreadGroup tg, String name)

→ This Constructor can be used to create Thread class object with the specified ThreadGroup name and Thread name.

Ex:-

ThreadGroup tg = new ThreadGroup("Java");

Thread t = new Thread(tg, "Core Java");

O/P:- Thread[Core Java, 5, Java].

(7) public Thread(ThreadGroup tg, Runnable r, String name)

→ This Constructor can be used to create Thread class object with the specified ThreadGroup name, Runnable name and Thread name.

Ex:-

Runnable r = new Thread();

ThreadGroup tg = new ThreadGroup("Java");

Thread t = new Thread(tg, r, "Core Java");

S.O.P() (t);

O/P:- Thread[Core Java, 5, Java].



### Methods:-

#### (1) setname() & getname()

→ setname() method can be used to set a particular name to the thread.

```
public void setName(String name)
```

getname() method can be used to get name of the thread.

```
public String getName()
```

Ex:-

```
Thread t = new Thread();
S.O.P(t.getName());
t.setName("AAA");
S.O.P(t.getName());
```

O/P:- Thread ->

AAA

#### (2) setPriority() & getPriority()

setPriority() method can be used to set a particular priority value to the thread.

```
public void setPriority(int Priority)
```

Where Priority value must be provided with in the range from 1 to 10. If we provide any priority value which is in outside range of 1 to 10 then JVM will raise an exception ~~like~~ that is `java.lang.IllegalArgumentException`.

getPriority() method can be used to get Priority value of the thread.

```
public int getPriority()
```

To represent priority values, Thread class has provided the following Constant variables.

```
public static final int MIN_PRIORITY = 1;
public static final int MAX_PRIORITY = 10;
public static final int NORM_PRIORITY = 5;
```

Ex:-

```
Thread t = new Thread();
System.out.println(t.getPriority()); → 5
t.setPriority(6);
System.out.println(t.getPriority()); → 6
t.setPriority(Thread.MAX_PRIORITY - 2);
System.out.println(t.getPriority()); → 10 - 2 = 8
OP:- 5
6
8
```

### ③ public boolean isAlive()

→ This method will check whether thread is in active or not.

Ex:-

```
Thread t = new Thread();
System.out.println(t.isAlive()); → false.
t.start();
System.out.println(t.isAlive()); → true.
OP:- false
true.
```

### ④ public static int activeCount()

→ This method will return an integer value representing no. of threads which are in active in present application.

Ex:-

```
Thread t = new Thread();
t.start();
System.out.println(Thread.activeCount());
OP:- 2 {Main Thread, User Thread}.
```

====

08/11/2015  
Sunday. (67)

### public static Thread currentThread()

→ This method can be used to return a Thread object which is active at present.

Eg:- class MyThread extends Thread

{

    public void run()

{

        for (int i=0; i<10; i++)

{

            System.out.println(Thread.currentThread().getName());

}

}

    class Test

    public static void main(String[] args){

        MyThread mt1 = new MyThread();

        MyThread mt2 = new MyThread();

        MyThread mt3 = new MyThread();

        mt1.setName("AAA");

        mt2.setName("BBB");

        mt3.setName("CCC");

        mt1.start();

        mt2.start();

        mt3.start();

}

z

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Public static void sleep( int time) throw InterruptedException

→ This method will ~~give~~ <sup>keep</sup> a running thread in Sleeping State upto the specified sleep time.

(Note:- In general, we will use sleep() method in run() method, where sleep() method will throws InterruptedException, here to handle InterruptedException we have to use try-catch-finally approach only, we are unable to use "throws" keyword in run() method prototype, because ○  
○ user defined Thread class we are overriding run() method, it must be same as java.lang.Thread class run() method, it is not having "throws" keyword in its prototype. ○  
○

Ex:-

Class MyThread extends Thread

```
{
 public void run()
 {
```

```
 try {
```

```
 for (int i=0; i<10; i++)
```

```
 Thread.sleep(1000);
```

```
 System.out.println("User Thread");
```

```
 } catch (Exception e)
```

```
 {
 e.printStackTrace();
 }
```

Notice we will take this value  
in ms (milliseconds)

```
class Test {
 public static void main (String [] args)
 {
 MyThread mt = new MyThread();
 mt.start();
 }
}
```

## ~~Short Test~~

(68)

public void join() throws InterruptedException

→ This method will pause a thread to complete another thread execution, when another thread executed completely then the paused thread will come back and it will complete its remaining execution part.

Ex:-

Class MyThread extends Thread

```

{
 public void run()
 {
 for (int i=0; i<10; i++)
 {
 System.out.println("User Thread");
 }
 }
}
```

Class Test {

```

 public void (String args)
 throws Exception
 {

```

```
 MyThread m2 = new MyThread();
```

```
 m2.start();
 m2.join();
```

```
 for (int i=0; i<10; i++)
```

```
 System.out.println("Main Thread");
```

public static void yield() throws InterruptedException

→ This method will pause a thread and giving control to some other thread available in waiting state having equal priority.

Note:- This method is not supported by windows OS, because this method must require priority values manipulations internally, which is not supported by Windows OS.

3

## Synchronization:-

In java applications, if we execute more than one thread on a single data item then that threads are called as Concurrent Threads, this process is called as Thread Concurrency.

In Threads Concurrency, if we execute more than one thread at a time then there may be a chance to get data inconsistency, it may not generate right results.

In the above context, to achieve data consistency we need a mechanism to allow only one thread, not to all the threads at a time.

In the above context, to provide data consistency JAVA has provided the above required mechanism called as Synchronization. Synchronization is a mechanism to allow only one thread at a time, not to allow more than one thread at a time, allowing other threads after completion of the present threads.

In java applications, synchronization is running on the basis of Locking mechanisms.

When we send more than one thread to synchronized part, lock manager will assign lock to highest priority thread, which thread acquire lock from lock manager that thread is eligible to execute synchronized part.

If a thread executed completely synchronized part then that thread may return lock to lock manager, where lock manager will assign lock to next thread.

In java applications, to perform synchronization JAVA has provided a keyword that is "synchronized" keyword.

In java applications we are able to provide synchronization in the following two ways:-

- ① Synchronized Methods
- ② Synchronized Blocks.
- ③ Synchronized Methods:-

(69)

If it's a normal java methods, a set of instructions, it able to allow only one thread at a time, it will allow other threads after completion of the present thread execution.

Ex:-

Class A

```

 {
 synchronized void m1()
 }

```

```

 for(;; i=0; i<10; i++)
 }

```

```

 System.out.println(Thread.currentThread().getName());
 }
}

```

Class MyThread2 extends Thread

```

{
 A a;
}

```

```

MyThread2(A a),
{

```

```

 this.a=a;
}

```

```

public void run()
{
}

```

```

 a.m1();
}
}

```

Class MyThread2 extends Thread

```

{
 A a;
}

```

```

MyThread2(A a)
{

```

```

 this.a=a;
}

```

```

public void run()
{
}

```

```

 a.m1();
}
}

```

Class MyThread3 extends Thread

```

{
 A a;
}

```

```

MyThread3(A a)
{

```

```

 this.a=a;
}

```

```

public void run()
{
}

```

```

 a.m1();
}
}

```

```

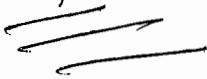
class Test {
 public void main(String[] args) {
 A a = new A();
 MyThread1 mt1 = new MyThread1(a);
 MyThread2 mt2 = new MyThread2(a);
 MyThread3 mt3 = new MyThread3(a);
 mt1.setName("AAA");
 mt2.setName("BBB");
 mt3.setName("CCC");
 mt1.start();
 mt2.start();
 mt3.start();
 }
}

```

Q. In java applications, to provide synchronization we have already synchronized methods then what is the requirement to used synchronized blocks?

→ In java apps, to provide synchronization, if we use synchronized methods then synchronization will be provided through out the ~~the~~ method irrespective of the actual requirement, it will reduce application performance.

In java apps, if we want to provide synchronization upto the required part in methods, not throughout the methods then we have to use synchronized blocks, this approach will improve application performance.



## SYNCHRONIZED SLOGAN:-

50

It is a set of instructions, it able to allow only one thread at a time, it will allow other threads after completion of the present thread execution.

## Syntax:-

Synchronized (Object of).

1

## instructions

1

CLASS A

۲

void m1()

1

```
S-O-Pin ("Before Synchronized Block:", &Thread.currentThread().
 getName());
```

Synchronized (this)

4

$\text{for}(\text{int } i=0; i<10; i++)$

1

```
S::ph("Inside Synchronized Block:", thread.currentThread()
 ()->getName());
```

( ).getname());

۱۷

class MyThread extends Thread → After that, as it is

10

A a ;

back program

~~Syncronized~~ Synchronized block program is same as synchronized method program but we have to use the above implementation for class A.

### Daemon Threads:-

Daemon threads are the threads, which are running internally in order to provide services for some other threads.

In java apps, Daemon threads must be terminated automatically along with a thread to which Daemon thread is providing services.

Ex:-

Garbage Collector is a daemon thread, it is running internally, it is providing services to the JVM and it will be terminated automatically when JVM is terminated.

To make any thread as daemon thread we have to use the following methods.

public void setDaemon(boolean b)

where if 'b' value is true then thread is daemon thread otherwise thread is not daemon thread.

To check any thread is daemon thread or not we have to use the following method.

public boolean isDaemon()

3

(71)

Ex:-

```

class MyThread extends Thread {
 public void run() {
 while(true) {
 System.out.println("User Thread");
 }
 }
}

class Test {
 public static void main(String[] args) {
 MyThread mt = new MyThread();
 mt.setDaemon(true);
 mt.start();
 System.out.println(mt.isDaemon());
 for(;; i=0; i<10; i++) {
 System.out.println("Main Thread");
 }
 }
}

```



SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

## Inter Thread Communication:-

The process of providing communication between more than one thread is called as Inter thread communication.

To provide inter thread communication in java apps we have to use the following methods from `java.lang.Object` class.

`public void wait()`

→ It will keep a running thread in waiting state.

`public void notify()`

→ It will give activation state to a thread which is available in waiting state.

`public void notifyAll()`

→ It will give activation state to all the threads which are available in waiting state.

Note:- All the above methods must be used under synchronization.

Inter threads communication will provide solutions for the problems like producer-consumer problem.

In producer-consumer problem, both producer and consumer are two threads, producer has to produce an item and consumer has to consume that item, some sequence has to managed upto infinite no. of times. Producer should not produce an item without consuming previous item by consumer and consumer must consume the item without that item is produced by producer.



Ex:-

```

class A {
 boolean flag = true;
 int count = 0;
 public synchronized void produce() {
 try {
 while (true) {
 if (flag == true) {
 count = count + 1;
 System.out.println("Producer produced:" + count);
 flag = false;
 notify();
 wait();
 } else {
 wait();
 }
 }
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
 public synchronized void consume() {
 try {

```

```
while(true)
{
 if(flag == free)
 {
 wait();
 }
 else
 {
 System.out("Consumer Consumed :" + count);
 flag = free;
 notify();
 wait();
 }
}
```

```
catch(Exception e)
{
 e.printStackTrace();
}
```

```
Class producer extends Thread
```

```
{
 A a;
 producer(A a)
 {
 this.a = a;
 }
 public void run()
 {
 a.produce();
 }
}
```

class Consumer extends Thread

(73)

```
A a;
Consumer(A a)
{
 this.a = a;
}
public void run()
{
 a.consume();
}

class Test
{
 public static void main(String[] args)
 {
 A a = new A();
 Producer p = new Producer(a);
 Consumer c = new Consumer(a);
 p.start();
 c.start();
 }
}
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### Deadlocks:-

Deadlock is a situation where more than one thread is depending on each other in circular dependency.

In java applications, Once if we are getting deadlock situation then we are unable to come out from deadlock situation, because, deadlock situation will provide structured exit state to the java program.

Deadlocks are not having recovery mechanisms, we have to use only prevention mechanism in java application.

Z

Ex:-

```
class Register_Course extends Thread
{
 Object course_name;
 Object faculty_name;
 Register_Course(Object course_name, Object faculty_name)
 {
 this.course_name = course_name;
 this.faculty_name = faculty_name;
 public void run()
 {
 synchronized(course_name)
 {
 System.out.println("Register Course thread holds course-name
 resource and waiting for faculty-name resource
 ");
 }
 synchronized(faculty_name)
 {
 System.out.println("Register Course thread holds both course-
 name and faculty-name resources,
 Registration is completed");
 }
 }
 }
}
```

Class Cancel-Course extends thread

{  
 Object course-name;  
 Object faculty-name;

Cancel-Course(Object C

    fwis. Course-Name = course-name;  
     fwis. faculty-name = faculty-name;

    public void run()

    {  
 synchronized (faculty-name)

        S.O.P("Cancel-Course Thread holds faculty-name  
 and waiting for course-name resource....");

    synchronized (~~faculty~~ course-name)

    S.O.P("Cancel-Course Thread holds both faculty-name  
     and course-name resources, so Cancellation is  
     Completed");

}

}

Object course-name = new Object();

Object faculty-name = new Object();

Register Course re-new Register-Course(Course-Name,  
faculty Name);

Cancel Course cc=new Cancel-Course(Course-

RC.start();

CC.start();

{ }

~~Multi threading finished~~

An array is an indexed Collection of fixed number of homogeneous data elements.

### Limitations of Object type Array:-

#### ② Arrays are fixed in size:-

Once we created arrays with some size there is no change of increasing or decreasing the size based on our requirement. Hence, to use arrays Concept Compulsarily we should know the size in advance, which may not be possible always.

#### ⑤ Arrays can hold Only Homogeneous datatype Element:-

Ex:- Student[] S = new Student[10000];  
S[0] = new Student(); ✓  
S[1] = new Customer(); ✗

Error:- Incompatible types found: Customer required Student  
We can Overcome this problem by using Object type arrays:-

Ex:- Object[] a = new Object[10000];  
a[0] = new Student(); ✓  
a[1] = new Customer(); ✓

#### ⑥ Arrays Concept isn't implemented based on some Standard Data Structure, so, readymade method support is not available. For every requirement programmer is responsible to write the code explicitly, which increases the complexity of program.

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

\* To overcome above problems of arrays we should go for collections.

→ Collections are Growable in nature, i.e. based on our requirement.  
We can increase and decrease the size.

→ Collections can hold both homogeneous and heterogeneous elements.  
→ Every Collection class is implemented based on some Standard data structure. Hence for every requirement ready-made method support is available. Being a programmer we can use this method directly and we are not responsible to implement this method.

### Differences between Arrays and Collections:-

#### Arrays

- ① Arrays are fixed in size.
- ② W.r.t. memory Arrays are not recommended to use.
- ③ W.r.t. performance, Arrays are recommended to use.
- ④ Array can hold only Homogeneous data type elements.
- ⑤ There is no Underline data structure for Arrays and hence ready-made ~~method~~ support is not available.
- ⑥ Array can hold both Primitives and Objects.

#### Collections

- ① Collections are Growable in nature.
- ② W.r.t. memory Collections are highly recommended to use.
- ③ W.r.t. performance, Collections are not recommended to use.
- ④ Collections can hold both Homogeneous and
- ⑤ Every Collection class is implemented based on some Standard data structure and hence ready-made method support is available for every requirement.
- ⑥ Collections can hold only Objects but not primitives.

## Collection :-

STL → Standard Template Library. 1

(P)

If you want to represent a group of individual objects as a single entity then we should go for Collection.

## Collection framework:-

It defines several classes and interfaces to represent a group of individual objects as a single entity.

java

→ Collection

→ Collection Framework

C++

→ Container

→ STL (Standard Template Library)

## 9 key interfaces of Collection framework:-

### ① Collection(I) :-

→ If we want to represent a group of individual objects as a single entity then we should go for Collection.

(\*)

≡

→ Collection interfaces defines the most common methods which are applicable for any collection object.

→ In general Collection interfaces considered as root interfaces of Collection framework.

Note: There is no Concrete class which implement Collection interface directly.

≡

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

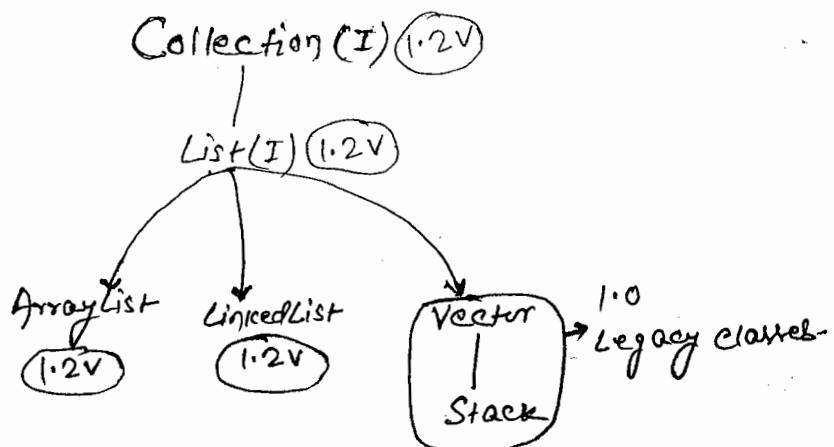
Ameerpet, Hyderabad.

## Collection vs Collections:-

- Collection is an interface, which can be used to represent a group of individual objects as a single entity.
- Collections is an Utility class to define several utility methods for Collection objects.

### ② List(I):-

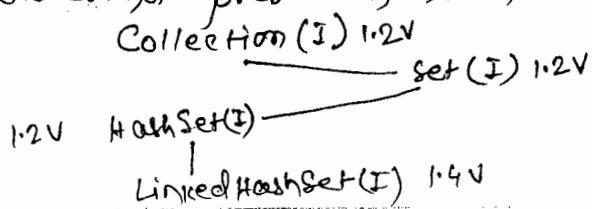
- It is child interface of collection.
- If we want to represent a group of individual elements, where duplicates are allowed and insertion order must be preserved then we should go for List.



Note:- In 1.2 version vector and stack classes are reengineered to implement List(I) interface.

### ③ Set(I):-

- It is the child interface of a collection.
- If we want to represent a group of individual objects as a single entity where duplicates are not allowed and insertion order not preserved, then we should go for Set(I).



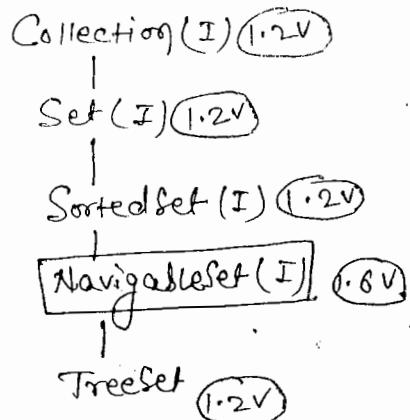
## Collection Set(I)

### (4) SortedSet(I):-

- It is the child interface of set.
- If we want to represent a group of individual objects as a single entity without duplicates according to some sorting order, then we should go for SortedSet(I).

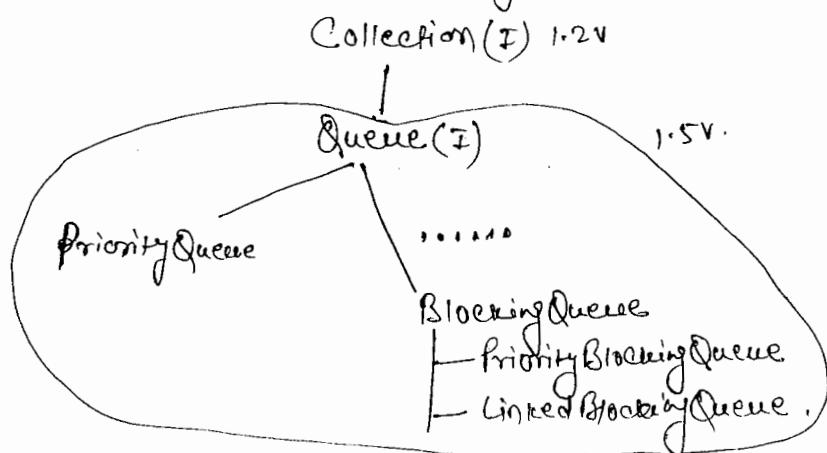
### (5) NavigableSet(I):-

- It is the child interface of SortedSet.
- It defines several methods for navigation purposes.



### (6) Queue(I):-

- It is the Child interface of collection.
- If we want to represent a group of individual objects prior to processing then we should go for Queue(I).

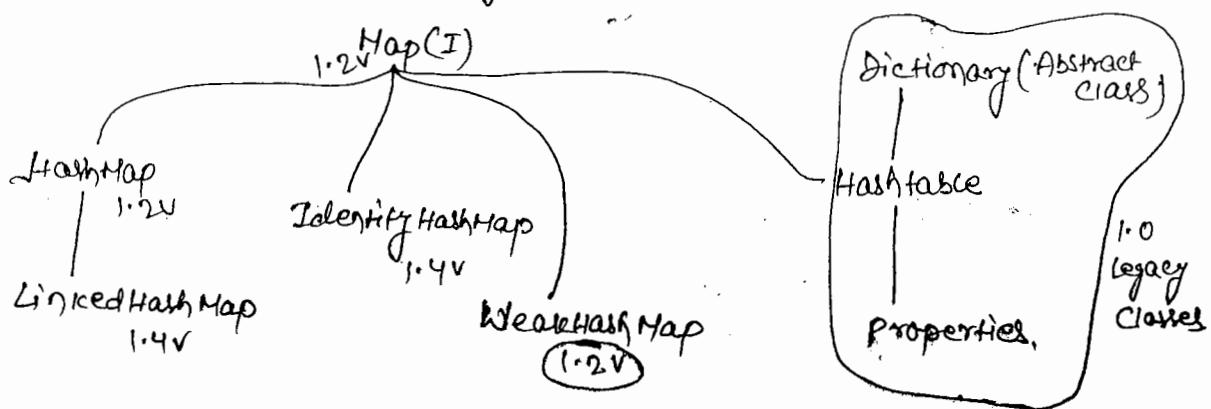


Note:- All the above interfaces (Collection, ListSet, SortedSet, NavigableSet, Queue) meant for representing individual objects as a single entity.

If we want to represent a group of objects as key-value pairs then we should go for Map.

#### (7) Map(I):-

- It is not child interface of Collection (I)
- If we want to represent a group of objects as key-value pairs, then we should go for Map(I).

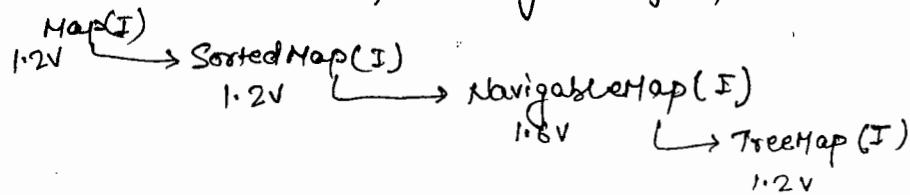


#### (8) SortedMap(I):-

- It is a child interface of Map(I).
- If we want to represent a group of key-value pairs according to some sorting order of keys, then we should go for SortedMap(I).

#### (9) NavigableMap(I):-

- It is the child interface of SortedMap(I).
- It defines several methods for navigation purposes.



Notes- In Collection framework the following are Legacy Characters. (78)

- ① Enumeration [E]
- ② Dictionary [Abstract Class]
- ③ Vector [C]
- ④ Stack [C]
- ⑤ Hashtable [C]
- ⑥ Properties [C]

### Sorting .

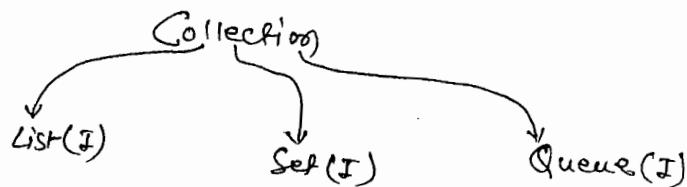
- ① Comparable
- ② Comparator

### Cursors

- ① Enumeration
- ② Iterator
- ③ ListIterator

### Utility Classes.

- ① Collections
- ② Arrays.



SRI RAGHAVENDRA XEROX

Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## Collection (I) :-

If we want to represent a group of individual objects as a single entity then we should go for Collection.

Most Common methods which are applicable for any Collection object.

Collection interface defines the following methods:-

- boolean add (Object o)
- boolean addAll (Collection C)
- boolean remove (Object o)
- boolean removeAll (Collection C)
- boolean retainAll (Collection C) → To remove all objects except those present in C.
- void clear()
- boolean contains (Object o)
- boolean containsAll (Collection C)
- boolean isEmpty()
- int size();
- Object[] toArray();
- Iterator iterator();

Note:- There is no Concrete class which implements Collection interface directly.

## List (I) :-

- It is the Child interface of Collection.
- If we want to represent a group of individual objects where duplicates are allowed and insertion order must be preserved then we should go for List (I).

(79)

We can differentiate duplicate values by using index and we can preserve insertion order by using index. Hence, index will play a very important role in List(I).

→ List(I) interface defines the following specific methods.

- void add(int index, Object o)
- boolean addAll(int index, Collection c)
- Object get(int index)
- Object remove(int index)
- Object set(int index, Object newObject)

→ To replace the element present at specified index with provided Object and returns Old object.

- if indexOf(Object o)
- returns index of first occurrence of 'o'.
- if lastIndexOf(Object o)
- ListIterator listIterator();

ArrayList:-

- ① The underlying data structure is resizable array or growable array.
- ② Duplicate Objects are allowed.
- ③ Insertion Order must be preserved.
- ④ Heterogeneous Objects are allowed (Except TreeSet and TreeMap every where Heterogeneous Objects are allowed).
- ⑤ Null insertion is possible.

Constructors:-

- ① ArrayList l = new ArrayList();  
Creates an empty ArrayList object with default initial capacity 10.  
Once ArrayList reaches its max capacity new ArrayList is created with  
New Capacity = (Current Capacity \* 3/2) + 1;

② `ArrayList l = new ArrayList(int initialCapacity);`

→ Creates empty array list object with the specified initial capacity.

③ `ArrayList l = new ArrayList(Collection c);`

→ Creates an equivalent ArrayList object for the given Collection.

This constructor mean for interconversion between Object.

Ex:-

```
import java.util.*;
class ArrayListDemo
{
 public static void main(String[] args)
 {
 ArrayList l = new ArrayList();
 l.add("A");
 l.add("10");
 l.add("A");
 l.add("null"); op:-
 System.out.println(l); // [A, 10, A, null]
 l.remove(2);
 System.out.println(l); // [A, 10, null]
 l.add(2, "M");
 l.add("N");
 System.out.println(l); // [A, 10, M, null, N].
 }
}
```

Usually, we can use collections to hold under transfer objects from one place to another place, to provide support for this requirement, every collection class implements Serializable and Cloneable interfaces.

ArrayList and Vector classes implements RandomAccess interface. So, that we can access any Random element with the same speed. Hence, ArrayList is the best choice if our frequent operation is retrieval operation.

### \* RandomAccess(I):-

→ Present in java.util package. Doesn't contain any method & hence it is Marker interface.

```
ArrayList l1 = new ArrayList();
LinkedList l2 = new LinkedList();
S.O.PIn(l1 instanceof Serializable); // true.
S.O.PIn(l2 instanceof Cloneable); // true
S.O.PIn(l1 instanceof RandomAccess); // true
S.O.PIn(l2 instanceof RandomAccess); // false
```

### Differences between ArrayList and Vector:-

- ① Every method present inside ArrayList is non-synchronized.  
Every method present inside Vector is synchronized.
- ② <sup>At a time</sup> Multiple Threads are allowed to operate on ArrayList object & it is not threadsafe.  
At a time, only one thread is allowed to operate on vector  
so it is thread safe.
- ③ Threads are not required to wait to operate on ArrayList object and hence relatively performance is high.  
Threads are required to wait to operate on Vector and hence relatively performance is low.

- ④ Introduced in 1.2v and it is ~~non~~-legacy.  
Introduced in 1.0v and it is legacy.

Q. How to get Synchronized Version of ArrayList Object?

→ By default ArrayList is non-synchronized, But we can get synchronized version of ArrayList Object by using `synchronizedList()` method of Collections class.

```
public static List synchronizedList(List l)
```

```
ArrayList l = new ArrayList();
```

```
List li = Collections.synchronizedList(l);
```

↳ Synchronized.

↳ non-synchronized

→ Similarly, we can find synchronized version of SetMap objects by using the following methods of Collections class.

```
public static Set synchronizedSet(Set s)
```

```
public static Map synchronizedMap(Map m)
```

Note:- ArrayList is the best choice if our frequent operation is retrieval operation.

ArrayList is the worse choice, if our frequent operation is insertion or deletion at the middle. (because internally several shift operations are required).

→ To handle this type of requirement, we should go for LinkedList.

## LinkedList(I):-

(8)

- ① The Underlying data Structure is ~~resizable~~ DoubleLinkedList
- ② Duplicates are allowed.
- ③ Insertion Order preserved.
- ④ Heterogeneous objects are allowed.
- ⑤ Null insertion is possible.
- ⑥ Best choice, if our frequent operation is insertion or deletion in the middle.
- ⑦ Worst choice, if our frequent operation is traversal operation.
- ⑧ Implements Serializable, Closeable but not RandomAccess interface.

## Constructors:-

`LinkedList l = new LinkedList();`

→ Creates an empty LinkedList Object.

`LinkedList l = new LinkedList(Collection c);`

→ Creates an equivalent LinkedList object for the given Collection.

Note:- Usually, we can use LinkedList to implements Stacks and Queues.

To provide support for this requirement. LinkedList class define the following six specific methods:-

`Void addFirst(Object o)`

`Void addLast(Object o)`

`Object getFirst()`

`Object getLast()`

`Object removeFirst()`

`Object removeLast()`

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Ex:-

```
import java.util.*;
Class LinkedListDemo{
 PSVM (String [] args){
 LinkedList l = new LinkedList();
 l.add ("deerga");
 l.add (30);
 l.add (null);
 l.add ("deerga");
 l.set(0, "software");
 l.add(0, "venky");
 l.removeLast();
 l.addFirst ("ccc");
 S.OPIN (l); // [Ccc, venky, software, 30, null]
 }
}
```

Vector :-

- ① The underlying data structure is resizable array and Growable array.
- ② Duplicates objects are allowed.
- ③ Insertion Order is preserved.
- ④ Null insertion is possible.
- ⑤ Heterogenous objects are allowed.
- ⑥ Implements Serializable, Cloneable and RandomAccess interfaces.
- ⑦ Best choice if our frequent operation is retrieval operation.
- ⑧ Worst choice if our frequent operation is insertion or deletion in the middle.
- ⑨ Every method is synchronized and hence Vector object is Thread-Safe ...

Constructors:-

① `Vector v = new Vector();`

Creates an empty vector Object with default initial capacity 10.  
Once vector reaches its max capacity, a new vector object will be created with double capacity.

② `Vector v = new Vector( int initialCapacity );`

③ `Vector v = new Vector( int initialCapacity, int incrementalCapacity );`

④ `Vector v = new Vector( Collection c );`

Vector Specific Methods:-To add elements:-

`add( Object o ) → C`

`add( int index, Object o ) → L`

`addElement( Object o ) → V`

To remove elements:-

`remove( Object o ) → C`

`removeElement( Object o ) → V`

`remove( int index ) → L`

`removeElement( int index ) → V`

`Clear() → C`

`removeAllElements() → V`

To get Objects:-

`Object get( int index ) → L`

`Object elementAt( int index ) → V`

`Object firstElement() → V`

`Object lastElement() → V`

## Other Methods:-

int size()

int capacity()

Enumeration elements()

Eg:-

```
import java.util.*;
class VectorDemo2{
 public static void main(String[] args){
 Vector v=new Vector();
 System.out.println(v.capacity()); // 10
 for(int i=1; i<=10; i++){
 v.addElement(i);
 }
 System.out.println(v.capacity()); // 10
 v.addElement("A");
 System.out.println(v.capacity()); // 20
 System.out.println(v); // [1, 2, ..., 10, A].
 }
}
```

## Stack:-

It is the child class of Vector.

It is specially designed class to implements LIFO (Last In First Out) operations.

## Constructors:-

```
Stack s=new Stack();
```

====

## Methods:-

Methods :-

Object push ( Object o )

to insert an object into the stack.

Object pop ()

to remove and return top of the stack.

Object peek ()

to return top of the stack without removal.

boolean empty ()

returns true if the stack is empty.

int search ( Object o )

returns offset if the element is available otherwise return -1.

Ex:-

```
import java.util.*;
class StackDemo{
 public static void main (String [] args){
 Stack s = new Stack ();
 s.push ("A");
 s.push ("B");
 s.push ("C");
 System.out.println (s);
 System.out.println (s.search ("A")); // 3
 System.out.println (s.search ("Z")); // -1
 }
}
```

index	offset
2	1
1	2
0	3

## The 3 cursors of java:-

If you want to get Objects one by one from the Collection, then we should go for cursor.

In java, there are three types of cursors are available.

- ① Enumeration
- ② Iterator
- ③ ListIterator.

### ① Enumeration:-

We can use enumeration to get the object one by one from the collection.

We can create enumeration objects by using elements() method of vector class.

public Enumeration elements();

Ex:- Enumeration e = v.elements();  
                  ↳ vector object.

### Methods:-

public boolean hasMoreElements();

public Object nextElement();

Ex:-  
import java.util.\*;  
class EnumerationDemo {  
    public static void main(String[] args) {  
        Vector v = new Vector();  
        for (int i=0; i<10; i++) {  
            v.addElement(i);  
        }  
        System.out.println(v); // [0, 1, 2, 3, ..., 10]  
        Enumeration e = v.elements();  
        while (e.hasMoreElements()) {

```

 Integer I = (Integer) e.nextElements();
 if (I%2 == 0)
 S.O.Println(l); // 0 2 4 6 8 10
 }
 S.O.Println(v); // [0, 1, 2, 3, ..., 10]
}

```

(8)

### Limitations of Enumeration:-

- ① We can apply enumeration concept only for legacy classes and it is not a universal cursor.
- ② By using enumeration we can perform only read operation and we can't perform remove operation.

To overcome above limitations, we should go for `Iterator(I)`.

### Iterator(I):-

- ① We can apply iterator(I) concept for any collection and hence it is universal cursor.
- By using iterator, we can perform both read and remove operations.

We can create Iterator object by using Iterator method of Collection interface.

`public Iterator iterator();`

CD:-

`Iterator itr = e.iterator();`

### Methods:-

```

public boolean hasNext();
public Object next();
public void remove();

```

3

Ex:-

```

import java.util.*;
class IteratorDemo {
 public static void main(String[] args) {
 ArrayList l = new ArrayList();
 for (int i=0; i<=10; i++)
 {
 l.add(i);
 }
 System.out.println(l); // [0, 1, 2, 3, 4, ..., 10]
 Iterator itr = l.iterator();
 while (itr.hasNext())
 {
 Integer I = (Integer) itr.next();
 if (I % 2 == 0)
 System.out.println(I); // 0 2 4 6 8 10
 else
 itr.remove();
 }
 System.out.println(l); // [0, 2, 4, 6, 8, 10]
 }
}

```

### Limitations of Iterator:-

- ① By using Enumeration and Iterator, we can always move only towards forward direction. And we can't move in backward directions. These are single directional Cursor but not bi-directional.
- By using

~~Replacement~~

To overcome these limitations, we should go for ~~ListIterator~~.

### ListIterator(I):-

By using ListIterator we can move either to the forward direction or, to the backward direction. i.e., it is a bi-directional cursor.

By using ListIterator we can perform replacement and addition of new Object in addition to read and remove operation.

We can create ListIterator Object by using ListIterator method or List(I) interface.

```
public ListIterator ListIterator()
```

e.g:- ListIterator ltr = l.ListIterator();  
       ↳ any list object.

ListIterator is a child interface of Iterator and hence all 3 methods of Iterator by default available to ListIterator. ListIterator defines the following 9 methods:-

```
public boolean hasNext()
public Object next()
public int nextIndex()
public boolean hasPrevious();
public Object previous();
public int previousIndex();
public void remove()
public void set(Object newObject)
public void add(Object newObject)
```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.



Eg:-

```
import java.util.*;
class ListIteratorDemo {
 public static void main(String[] args) {
 LinkedList l = new LinkedList();
 l.add("Sunny");
 l.add("Venki");
 l.add("Chiru");
 l.add("Nag");
 System.out.println(l); // [Sunny, Venki, Chiru, Nag]
 ListIterator ltr = l.listIterator();
 while(ltr.hasNext()) {
 String s = (String) ltr.next();
 if(s.equals("Venki"))
 {
 ltr.remove(); // [Sunny, Chiru, Nag]
 }
 else if(s.equals("Nag"))
 {
 ltr.add("Chaitu"); // [Sunny, Chiru, Nag, Chaitu]
 }
 else if(s.equals("Chiru"))
 {
 ltr.set("Charan"); // [Sunny, Charan, Nag, Chaitu]
 }
 }
 System.out.println(l); // [Sunny, Charan, Nag, Chaitu]
 }
}
```

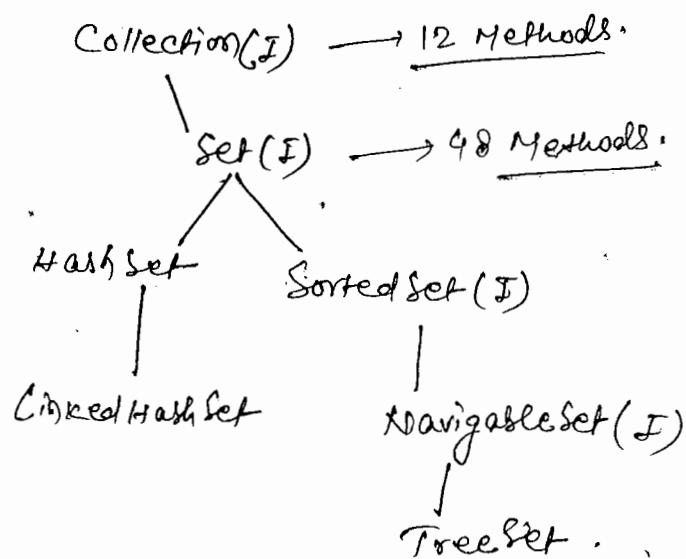
Note:- The most powerful Cursor is list iterator. But its limitation is ~~that~~ it's applicable only for list object.

### Comparison table of 3 Cursors in java:-

Properties	Enumeration	Iterator	ListIterator
① Where it is applicable?	Only for legacy class.	for any collection object.	for any list object.
② Movement	Single direction (Only forward)	Single direction (Only forward)	Bidirectional
③ How to get?	By using elements() method of vector class.	By using iterator() method of collection interface.	By using listIterator() method of List(I) interface
④ Operations can be performed	Only Read	Both Read and Remove	Read, Remove, Replace and add.
⑤ Methods	2 methods hasMoreElements() nextElement()	3 Methods hasNext() next() remove()	9 methods.
⑥ Is it legacy	Yes (1.0v)	No (1.2v)	No (1.2v)

III

## Set(I) :-



- Set is child interface of Collection.
- If we want to represent a group of individual objects where duplicates are not allowed and insertion order is not preserved, then we should go for set.
- Set(I) interface doesn't contain any new method and we have to use only collection interface methods.

### HashSet :-

- The underline data structure is hashtable.
- Duplicates object are not allowed.
- If we are trying to insert duplicate objects, we won't get any compile time and runtime errors. & that method returns "false".
- Insertion order is not preserved. And it is based on HashCode Object.
- Null insertion is possible (Only once).
- Heterogeneous Objects are allowed.

- Implements Serializable, Cloneable, but not the RandomAccess.
- HashSet is the best choice if our frequent operation is "Search Operation".

### Constructors:-

→ HashSet h = new HashSet();

Create empty HashSet object with default initial capacity 16 & default fill ratio 0.75.

HashSet h = new HashSet (int initialCapacity, float fillRatio);

HashSet h = new HashSet(Collection c)

Create an empty HashSet (~~int initialCapacity~~) object with specified initial capacity & default fill ratio is 0.75.

HashSet h = new HashSet (int initialCapacity, float fillRatio)

HashSet h = new HashSet(Collection c)

### Load factor/fill ratio:-

After loading how much factor a new HashSet object is required to create, that factor is called Loadfactor or Fill Ratio.

Ex:- fill Ratio 0.75 means after filling 75% Ratio a new HashSet object is required to create.

Ex:-

```
import java.util.*;
class HashSetDemo {
 public static void main(String[] args) {
 HashSet h = new HashSet();
 h.add("B");
 h.add("C");
 h.add("Z");
 h.add(null);
 h.add(10);
 System.out.println(h.add("Z")); // false
 System.out.println(h); // [null, 0, B, C, 10, Z]
 }
}
```

### LinkedHashSet-

- It is a child class of HashSet.
- It is exactly same as HashSet (including methods and constructors) except some following differences.)

### Difference between HashSet and LinkedHashSet :-

- ① The underline data structure is Hashtable. → HashSet  
Underline data structure is a combination of UnorderedList and Hashtable. → LinkedHashSet.
- ② Insertion Order is not preserved  
Insertion Order is preserved.
- ③ Introduced in 1.2V  
Introduced in 1.4V.  
≡

In the above program if we replace HashSet with LinkedHashMap  
 - AshSet, then the output is [B, C, D, Z, null, 10]. that is  
 insertion Order is preserved.

Note:- In general, we can use LinkedHashSet and LinkedHash  
 - Map for developing Cache based applications. where, duplicates  
 are not allowed under insertion order must be preserved.

### SortedSet(T):

- It is the Child interface of Set.
- If we want to represent a group of individual Objects without duplicates according to some Sorting Order then we should go for SortedSet.
- The Sorting Order can be either default natural sorting order or Customized Sorting Order.
- The default natural sorting order for numbers is ascending order whereas default natural sorting order for String objects is alphabetical Order.
- SortedSet(T) interface defines the following Six Specific methods.

### Object first();

returns first element of the SortedSet.

### Object last();

returns last element of the SortedSet.

### SortedSet headSet (Object obj);

returns SortedSet whose elements are less than obj

### SortedSet tailSet (Object obj);

return SortedSet whose elements are  $\geq$  obj

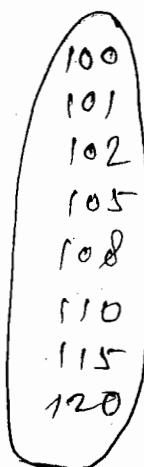
### SortedSet subset(Object obj1, Object obj2)

Returns SortedSet where elements are  $\geq$  obj1 and  $<$  obj2.

### Comparator comparator()

Returns Comparator object that describes underlying sorting technique. if we are using default natural sorting order then we will get null.

- ① first()  $\rightarrow$  100
- ② last()  $\rightarrow$  120
- ③ headSet(105)  $\rightarrow$  [100, 101, 102]
- ④ tailSet(105)  $\rightarrow$  [105, 108, 110, 115, 120]
- ⑤ subSet(102, 115)  $\rightarrow$  [102, 105, 108, 110]
- ⑥ Comparator()  $\rightarrow$  null



### TreeSet:-

- The Underlying data Structure is Balanced Tree
- Duplicate objects are not allowed.
- Insertion Order is not preserved as it's based on some sorting Order.
- Heterogeneous objects are not allowed, Otherwise we will get Runtime Exception ~~Same~~ ClassCastException.
- Null insertion is possible (Only once).
- Implements Serializable, Comparable interface but not RandomAccess.

## Constructors :-

- ① TreeSet t = new TreeSet();
- Creates an empty TreeSet Objects where the elements will be inserted according to default natural sorting order.
- ② TreeSet t = new TreeSet(Comparator c);
- Creates an empty TreeSet Object where the elements will be inserted according to Customized Sorting specified by Comparator object.
- ③ TreeSet t = new TreeSet(SortedSet s)
- ④ TreeSet t = new TreeSet(Collection c);

Ex:-

```

import java.util.*;
class TreeSetDemo{
 public static void main(String[] args){
 TreeSet t = new TreeSet();
 t.add("A");
 t.add("a");
 t.add("B");
 t.add("Z");
 t.add("L");
 //t.add(new Integer(10)); // CCE
 //t.add(null); // -NPE
 System.out.println(t);
 }
}

```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

O/P:- [A, B, L, Z, a].

—E

### null acceptance :-

- for non-empty TreeSet if we are trying to insert null then we will get NullPointerException.
- for empty TreeSet at the first element null insertion is possible but after inserting that null if we are trying to insert any other element then we will get NullPointerException.  
→ ~~too~~.

(Note:- from 1.7v onwards, even at first element null to the empty TreeSet is not allowed otherwise we will get runtime exception same as NullPointerException.

but upto 1.6v for empty TreeSet at the first element null is allowed.)

\*\* → If we are depending on default naturals of the Order then Compulsory the object should be homogeneous and Comparable. Otherwise, we will get Runtime Exception:- ClassCastException. \*\*

- An object is said to be Comparable if and only if the corresponding class implements java.lang.Comparable interface.
- All wrapper classes and String Classes already implements Comparable interface but StringBuffer doesn't implement Comparable interface.

Ex:-

```
import java.util.*;
class TreeSetDemo1 {
 public static void main (String [] args) {
 TreeSet t = new TreeSet();
 t.add (new StringBuffer ("A"));
 t.add (new StringBuffer ("Z"));
 t.add (new StringBuffer ("L"));
 t.add (new StringBuffer ("B"));
 System.out.println (t);
 }
}
```

Opp:- ClassCastException  
as StringBuffer Object  
doesn't contain Comparable  
interface i.e.  
Cannot be compared.

## Comparable Interface :-

- Present in `java.lang` package.
- It contains only one method `compareTo()`.

`public int compareTo(Object obj)`

`Obj1.compareTo(Obj2)` → if and only if.

→ returns -ve iff Obj1 has to come before Obj2.

→ returns +ve iff Obj1 has to come after Obj2.

→ returns 0 iff Obj1 and Obj2 are equal.

Ex:-

`Class Test{`

`PSVM (String[] args){`

`S.O.PIn ("A".compareTo("Z"));` // -25 → +ve

`S.O.PIn ("Z".compareTo("K"));` // 15 → -ve

`S.O.PIn ("A".compareTo("A"));` // 0 → 0

`S.O.PIn ("A".compareTo(null));` // → NullPointerException.

(Note:- The object which we are trying to insert is Obj1.  
The object which is already inserted is Obj2)

`Obj1.compareTo(Obj2)`

→ If we are depending on default natural sorting Order, then while inserting objects to the TreeSet, JVM will call `compareTo()` method, to implement sorting.

Ex:-

`TreeSet t = new TreeSet();`

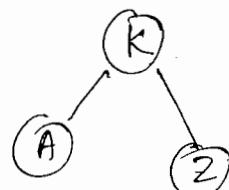
~~`t.add("K");` → "K" is greater than "A", "Z".~~

`t.add("A");` → "A".compareTo("K") → -ve

`t.add("Z");` → "Z".compareTo("K") → +ve

`t.add("K");` → "K".compareTo("K") → 0

`S.O.PIn(t);` // [A, K, Z]



In-Order traversal  
Left-Root-Right

Note:- If we are not satisfy with default natural sorting order or, if the default natural sorting order not already available then we can define our own sorting order by using Comparator.

→ Comparable meant for default natural sorting order whereas,

Comparator meant for Customize Sorting Order.

### Comparator:-

15/11/2015

→ Present in java.util package

→ Defines the following two methods.

① public int compare(Object obj1, Object obj2)

→ return -ve iff obj1 has to come before obj2

→ return +ve if obj1 has to come after obj2

→ returns 0 iff obj1 and obj2 are equal.

② public boolean equals (Object obj)

\* whenever we are implementing Comparator interface we have to provide implementation only for Compare method.

\* Implementing equal method is Optional, because it is already available from object class through Inheritance.

(91)

Q. WAP to insert integer objects into the TreeSet, where the elements will be inserted according to descending order.

Ex:-

```
import java.util.*;
class TreeSetDemo3 {
 public static void main(String[] args) {
 TreeSet t = new TreeSet(new MyComparator());
 t.add(10);
 t.add(0);
 t.add(15);
 t.add(5);
 t.add(20);
 t.add(20);
 System.out.println(t);
 }
}
```

Class MyComparator implements Comparator

```
{
 public int compare(Object obj1, Object obj2) {
 Integer I1 = (Integer) obj1;
 Integer I2 = (Integer) obj2;
 if (I1 < I2)
 return +1;
 if (I1 > I2)
 return -100;
 else
 return 0;
 }
}
```

O/P :- [20, 15, 10, 5, 0].

≡

TreeSet t = new TreeSet(new MyComparator()); → ①

t.add(10); ✓

t.add(0); ↗ Compare(0, 10);

t.add(15); ↗ Compare(15, 10);

t.add(5); ↗ Compare(5, 10)

t.add(20); ↗ Compare(5, 0)

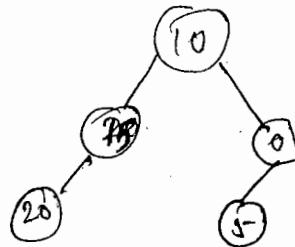
t.add(20); ↗ Compare(20, 10)

t.add(20); ↗ Compare(20, 15)

t.add(20); ↗ Compare(20, 10)

0 ↗ Compare(20, 15)

0 ↗ Compare(20, 20)



→ At line 1 if you are not passing Comparator object internally JVM will call compareTo() method, which is meant for default natural sorting order (Ascending Order). In this case Output is [0, 5, 10, 15, 20].

→ At lines if you are passing comparator object then JVM will call compare() method which is meant for Customized sorting (descending order). In this case Output is [20, 15, 10, 5, 0].

→ Various possible implementations of compare() method :-

```
public int compare(Object obj1, Object obj2)
{
```

```
 Integer I1 = (Integer) obj1;
```

```
 Integer I2 = (Integer) obj2;
```

```
 return I1.compareTo(I2); // [5, 10, 15, 20] default natural sorting order.
```

(4) (92)

```

return -I1.compareTo(I2); // [20, 15, 10, 5, 0] reverse of default
 natural sorting order
return I2.compareTo(I1); // [20, 15, 10, 5, 0] reverse of default
 natural sorting order.
return -I2.compareTo(I1); // [0, 5, 10, 15, 20] default natural sorting
 order
return +1; // [
return -1;
return 0; // Only first inserted
]

```

Q.

```

⇒ import java.util.*;
class TreeSetDemo2{
 public static void main(String[] args){
 TreeSet t = new TreeSet(new MyComparator());
 t.add("Raja");
 t.add("Shobhanani");
 t.add("Rajakumari");
 t.add("GangaBharani");
 t.add("Kamalamma");
 System.out.println(t);
 }
}

```

```

CLASS MyComparator implements Comparator{
 public int compare(Object obj1, Object obj2)
 {
 String s1 = (String) obj1;
 String s2 = obj2.toString();
 }
}

```

```
//return -s1.compareTo(s2);
return s2.compareTo(s1);
}
```

O/P:- [Shobhalakshmi, Roja, Ramulamma, Rajakumari, Gangabharani].

Q. WAP to insert StringBuffer object into the TreeSet according to alphabetical Order.

```
import java.util.*;
class TreeSetDemo3
{
 public static void main(String[] args)
 {
 TreeSet t = new TreeSet(new MyComparator());
 t.add(new StringBuffer("A"));
 t.add(new StringBuffer("Z"));
 t.add(new StringBuffer("K"));
 t.add(new StringBuffer("L"));
 System.out.println(t); // [A, K, L, Z]
 }
}
```

Class myComparator implements Comparator

```
{
 public int compare(Object obj1, Object obj2)
 {
 String s1 = obj1.toString();
 String s2 = obj2.toString();
 return s1.compareTo(s2);
 }
}
```

(Q1)

Q. WAP to insert String under StringBuffer Object into TreeSet where the sorting order is increasing length order. If two objects having the same length then consider their alphabetical Order.

Ans:-

```
import java.util.*;
class TreeSetDemo{
 public static void main(String[] args){
 TreeSet t = new TreeSet(new MyComparator());
 t.add("f");
 t.add(new StringBuffer("ABC"));
 t.add(new StringBuffer("AA"));
 t.add(new StringBuffer("xx"));
 t.add("ABCD");
 t.add("g");
 System.out.println(t); // [A, AA, xx, ABC, ABCD].
 }
}
```

Class MyComparator implements Comparator

```
{
 public int compare(Object obj1, Object obj2){
 String s1 = obj1.toString();
 String s2 = obj2.toString();
 int i1 = s1.length();
 int i2 = s2.length();
 if (i1 < i2)
 return -1;
 else if (i1 > i2)
 return 1;
 else
 return s1.compareTo(s2);
 }
}
```

Note:- If we are depending on default natural sorting Order, then objects should be homogeneous and Comparable, Otherwise we will get runtime exception saying Class Cast Exception.

If are defining our own sorting by Comparator then Objects need not be homogeneous and Comparable. We can <sup>add</sup> heterogeneous non-comparable Objects also.

### Comparable vs. Comparator:-

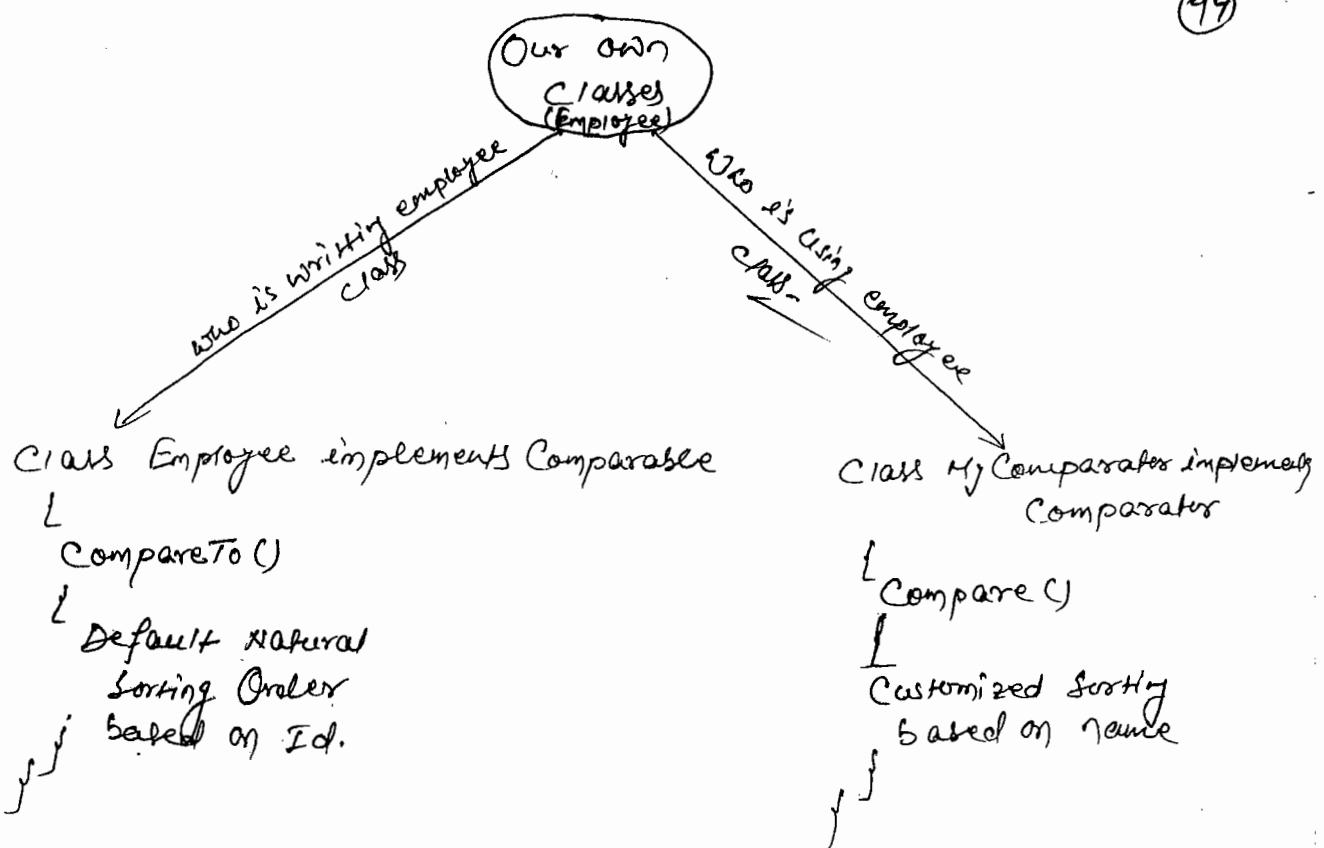
① for predefined Comparable classes, default natural sorting Order already available. If we are not satisfied with that we can define our own sorting by using Comparator.

Eg:- String.

② for Predefined non Comparable classes, (like StringBuffer) default natural sorting order not already available. We can define our own sorting by using Comparator.

③ for our own classes (like employee), the person who is writing employee class, is responsible to implement default natural sorting order by implementing Comparable interface. The person who is using employee class, he is responsible to implement customize sorting by implementing Comparator. (If it is not satisfied with default natural sorting Order).





Ex:-

```

import java.util.*;
class Employee implements Comparable
{
 String name;
 int eid;
 Employee (String name, int eid)
 {
 this.name = name;
 this.eid = eid;
 }
 public String toString()
 {
 return name + " " + eid;
 }
 public int compareTo (Object obj)
 {
 int eid1 = this.eid;
 Employee e = (Employee) obj;
 int eid2 = e.eid;
 if (eid1 < eid2)
 return -1;
 else if (eid1 > eid2)
 return 1;
 else
 return 0;
 }
}

```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

```

 {
 return -1;
 }
 else if (eid1 > eid2)
 {
 return 1;
 }
 else return 0;
}

```

### Class CompComp

```

class CompComp
{
 PSVM(String[] args){

```

```

Employee e1 = new Employee("nag", 100),

```

```

Employee e2 = ("Balaiab", 200);

```

```

e3 = ("Chiru", 50);

```

```

e4 = ("Venki", 150);

```

```

e5 = ("nag", 100);

```

```

TreeSet t = new TreeSet();

```

```

t.add(e1);

```

```

t.add(e2);

```

```

t.add(e3);

```

```

t.add(e4);

```

```

t.add(e5);

```

```

System.out.println(t); // [Chiru-50, nag-100, Venki-150, Balaiab-200]

```

TreeSet

Continued... ]

(95)

```
TreeSet t1 = new TreeSet(new MyComparator());
```

```
t1.add(e1);
t1.add(e2);
t1.add(e3);
t1.add(e4);
t1.add(e5);
t1.add(e6);
```

s.o.println(t1); // [Balaji-200, Chiru-50, Nag-100, Venki-150].

{}

class MyComparator implements Comparator

{ public int compare(Object obj1, Object obj2)

{

```
Employee e1 = (Employee) obj1;
e2 = (Employee) obj2;
```

```
String s1 = e1.name;
```

```
String s2 = e2.name;
```

```
return s1.compareTo(s2);
```

{}

① It is meant for default nature of sorting Order  
It is meant for Customized sorting Order.

② Present in java.lang package

present in java.util package

③ It defines only one method compareTo()

It defines two methods compare() and equals()

④ All wrapper classes are String class implements Comparable interface.

The only implemented classes of Comparator are Collator or RuleBasedCollator.

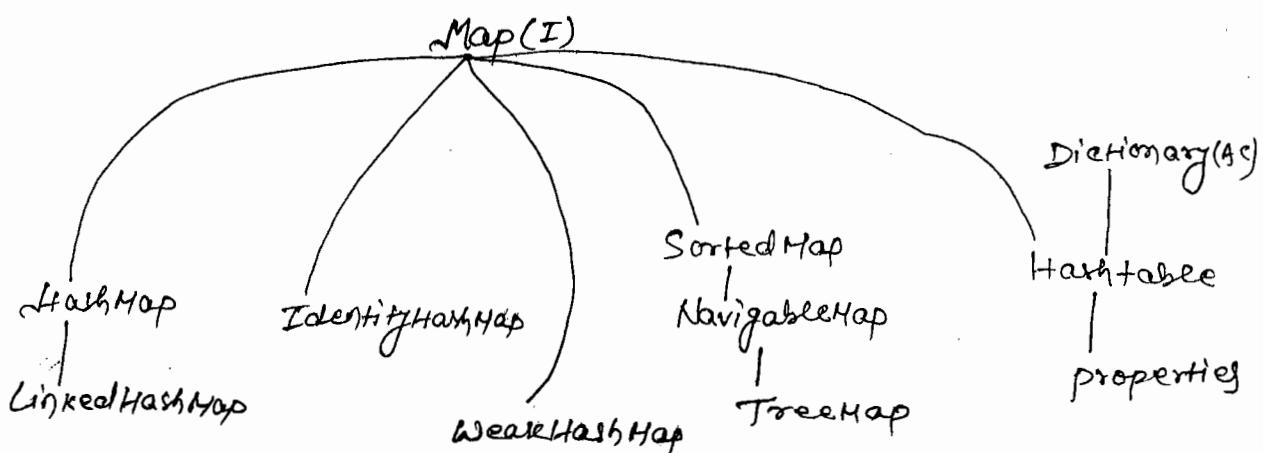
Comparison table of Set Implemented Classes

Properties	HashSet	LinkedHashSet	TreeSet
① Underlying Data Structure	HashTable	LinkedList + HashTable	Balanced Tree
② Insertion Order	Not Preserved	Preserved	Not Preferred
③ Sorting Order	NOT Applicable	NOT Applicable	Applicable
④ Duplicate Objects	NOT Allowed	NOT Allowed	NOT Allowed
⑤ Heterogeneous Object	Allowed	Allowed	NOT Allowed
⑥ Null insertion	Allowed Only Once	Allowed Only once	for empty set at a first element null is allowed otherwise we will get NullPointerException from 1.7v it is also not allowed
⑦ Introduced in	1.2v	1.4v	1.2v



## Map(I) :-

(96)



→ Map is not child interface of Collection.

→ If we want to represent a group of objects as key-value pairs, then we should go for Map(I).

rollno — name  
 mobile number — Address  
 ipaddress — domain name.

→ Duplicate keys are not allowed but values can be duplicated.

→ Each key-value pair is called an "Entry"

## Map interface methods :-

① Object put(Object key, Object value)

→ To insert one key-value pair.

→ If the key is already available then Old value will be replaced with new value and returns Old value.

Eg:-

null - m.put(101, "durga");

null - m.put(102, "shiva");

durga - m.put(101, "kanjan");

3

Key	values
101	Durga
102	Ravi
103	Shiva
104	Patwon

② void putAll(Map m)

③ Object get(Object key)

→ returns the value associated with specified key.

④ Object remove(Object key)

remove the entry associated with specified key.

⑤ boolean containsKey(Object key)

⑥ boolean containsValue(Object value)

⑦ boolean isEmpty()

⑧ int size()

⑨ void clear();

⑩ Set keySet()

⑪ Collection values()

⑫ Set entrySet()

} Collection views of Map.

### Collection views of Map:-

#### Entry (I):-

→ Each key-value pair is called an Entry.

→ Without existing Map Object there is no chance of existing entry object.

→ Hence interface Entry is defined inside Map entries.

Ex:-

interface Map

  └ interface Entry

    Object getKey()

    Object getValue()

    Object setValue(Object newValue)

## HashMap:

(97)

- The underlying data structure is
- Insertion order is not preserved and it is based on hashCode of keys.
- Duplicate keys are not allowed but values can be duplicated.
- Heterogeneous objects are allowed for both key and value.
- Null is allowed for key, Only once.
- Null is allowed for value (Any no. of times)
- Implements Serializable, Cloneable, but not RandomAccess.
- Best choice if our frequent operation is search operation.

## Constructors:-

```
HashMap m = new HashMap();
```

- Created an empty HashMap Object with default initial capacity 16 and default fill ratio is 0.75.

```
HashMap m = new HashMap(int initialCapacity);
```

```
HashMap m = new HashMap(int initialCapacity, float fillratio);
```

```
HashMap m = new HashMap(Map m);
```

Ex:-

```
import java.util.*;
class HashMapDemo{
 public static void main(String[] args){
 HashMap m = new HashMap();
 m.put("Chiranjivi", 700);
 m.put("Balaji", 800);
 m.put("Venkatesh", 200);
 m.put("Nagarjuna", 500);
 System.out.println(m); // {k=v, k=v, ...}
 System.out.println(m.put("Chiranjivi", 1000)); // 700
 Set s = m.keySet();
 System.out.println(s); // {k, k, ...}
```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

```

Collection c = m.values();
s.o.pn(c); // [v, v, v, ...]

Set s1 = m.entrySet();
s.o.pn(s1); // [k=v, k=v; ...]

Iterator itr = s1.iterator();
while(itr.hasNext())
{
 Map.Entry m1 = (Map.Entry)itr.next();
 s.o.pn(m1.getKey() + " - " + m1.getValue());
 if(m1.getKey().equals("nagarjuna"))
 {
 m1.setValue(10000);
 }
 s.o.pn(m1);
}

```

### Differences b/w HashMap and Hashtable:-

<u>HashMap</u>	<u>Hashtable</u>
① No method in HashMap is synchronized.	① Every method present inside Hashtable is sync.
② At a time Multiple threads are allowed to operate on HashMap object and hence HashMap is not thread safe.	② At a time, Only One thread is allowed to operate on Hashtable and Hashtable object is thread safe.
③ Threads are not required to wait to operate on HashMap object and hence relatively performance is high.	③ Threads are required to wait to operate on hashtable and hence relatively performance is low.

### HashMap

- \* ④ Null is allowed for both key and values.

- ⑤ Introduced in 1.2v.

### Hashtable

(78)

- \* ④ Null is not allowed for both keys and values, Otherwise we will get NullPointerException

- ⑤ Introduced in 1.0v.

### How to get Synchronized Version of HashMap Object:-

→ By default, HashMap is non-synchronized, but we can get synchronized version by using synchronizedMap() method of Collection class.

Ex:- `HashMap m = new HashMap();`

`Map m1 = Collections.synchronizedMap(m);` → Non-synchronized  
↳ Synchronized.

### LinkedHashMap:-

→ It is the child class of HashMap.

→ It is exactly seems as an HashMap except the following differences.

### HashMap

- ① The underlying data structure is Hashtable

- ② Insertion Order is not preserved.

- ③ Introduced in 1.2v.

### LinkedHashMap

- ① The underlying data structure is Hashtable + LinkedList

- ② Insertion Order is preserved.

- ③ Introduced in 1.4v

3

3

→ In the above example, if we replace HashMap with the LinkedHashMap, then the output is

{ Chiranjivi = 700, Balaiyah = 800, Venkatesh = 200, Nagarjuna = 500 }  
i.e. insertion order is preserved.

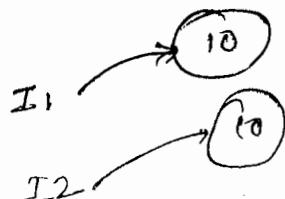
Note:- In general, we can use a LinkedHashSet and LinkedHashMap to implement Cache-based applications.]

Difference b/w == operator and equals() method :-

- In general, we can use == operator, for reference comparison.  
i.e. if two references pointing to the same object then  
~~double~~ == operator, returns true.
- In general we can use .equal() method for contained comparison.  
i.e. even though Objects are different if the content is same  
then we will get true.

Ex:-

```
Integer I1 = new Integer(10)
Integer I2 = new Integer(10)
S.O.Pn(I1 == I2); // false
S.O.Pn(I1.equals(I2)); // true
```



IdentityHashMap :-

- It is exactly same as HashMap except the following diff.
- In the case of HashMap JVM will use .equals() method to identify duplicate keys, which is meant for content comparison.
- In IdentityHashMap JVM will use == operator to identify duplicate keys, which is meant for reference comparison.

(99)

Ex:-

```

HashMap m = new HashMap();
Integer i1 = new Integer(10);
Integer i2 = new Integer(10);
m.put(i1, "pawan");
m.put(i2, "Kalyan");
System.out(m);
}
}

```

*i1 and i2 are duplicate keys, bcz i1 == i2 returns false.*

In the above code, if we replace HashMap with IdentityHashMap, then i1 and i2 are not duplicate keys (becz,  $i1 == i2$  returns false).

In this case output is  
 $10 = \text{pawan}$   
 $10 = \text{Kalyan}$ .

### WeakHashMap:-

- It is exactly seen as HashMap, except the following difference.
- In the case of Normal HashMap, Even though Object doesn't contain any reference, still it is not eligible for "gc". if it is associated with HashMap i.e. HashMap dominates 'gc' (garbage collector).
- But in the case of WeakHashMap, if the Object doesn't contain reference then it is always eligible for "gc", even though it is associated with \*.

Eg:-

```
import java.util.*;
class WeakHashMapDemo{
 public String(Temp) throws Exception
{
 HashMap m = new HashMap();
 Temp t = new Temp();
 m.put(t, "durga");
 System.out.println(m); // {temp=durga}
 t=null;
 System.out.println(m); // {temp=durga}
}
```

Class Temp

```
{
 public String toString()
 {
 return "temp";
 }
 public void finalize()
 {
 System.out.println("Finalize method called");
 }
}
```

OP:-

In the above example if we replace HashMap with the  
~~of~~ <sup>Weak</sup>HashMap then Object won't be destroyed by the garbage  
Collector in this the output is {Temp=durga}

{Temp=durga}

## SortedMap:-

(100)

- If we want to represent a group of key-value pairs according to some sorting order, then we should go for SortedMap.
- The sorting can be either default natural sorting order or Customized sorting order.
- Sorting is always based on the keys but not on the values.
- Sorted Map defines the following six specific methods:
  - \* Object firstKey();
  - \* Object lastKey();
  - \* SortedMap headMap(Object key)
  - \* SortedMap tailMap(Object key)
  - \* SortedMap subMap(Object key, Object key2)
  - \* Comparator comparator()

## TreeMap:-

- Underlying data structure is Red-Black Tree.
- Insertion Order is not preserved and it is based on some sorting order of keys.
- Duplicate keys are not allowed but values can be duplicated.
- If we are depending on default natural sorting order then keys should be homogeneous and Comparable; otherwise we will get ClassCastException.
- If we are defining our own sorting by comparator then the keys need not be homogeneous and Comparable.
- But, there are no restrictions on values, they can be heterogeneous and non-Comparable.

Z

### Null acceptance:-

- for non-empty TreeMap, if we are trying to insert an entry with null key, then we will get NullPointerException.
- for empty TreeMap as the first entry with the null key is allowed but after inserting that entry if we are trying to insert any other entry then we will get NullPointerException.
- There are no restrictions of using null for values. We can use any no. of times and anywhere.

Note:- from 1.7 onwards even for empty TreeMap as the first entry with null key also not allowed but no restrictions for values.]

### Constructors:-

TreeMap t = new TreeMap();

→ for default Natural Sorting Order.

TreeMap t = new TreeMap(Comparator c);

→ for Customized sorting order

TreeMap t = new TreeMap(SortedMap m);

TreeMap t = new TreeMap(Map m);

Ex:-

```
import java.util.*;
class TreeMapDemo3{
 public static void main(String[] args){
 TreeMap m = new TreeMap();
 m.put(100, "222");
 m.put(108, "yyy");
 m.put(101, "xxx");
 m.put(104, 106);
 //m.put("AAA", "XXX"); //CE
 //m.put(null, "XXX"); //NPE
 System.out.println(m); // 100=222, 101=XXX,
 // 103=yyy, 104=106
 }
}
```

Ex:-

(PPT)

```
import java.util.*;
class TreeMapDemo{
 public static void main(String[] args){
 TreeMap t = new TreeMap(new MyComparator());
 t.put("XXX", 10);
 t.put("AAA", 20);
 t.put("222", 30);
 t.put("LLL", 40);
 System.out.println(t); // { 222=30, XXX=10, LLL=40, AAA=20 }
 }
}
```

```
class MyComparator implements Comparator{
 public int compare(Object obj1, Object obj2){
 String s1 = obj1.toString();
 String s2 = obj2.toString();
 return s2.compareTo(s1);
 }
}
```

### Hashtable:-

- The underlying data structure is hashtable.
- Duplicate keys are not allowed, but the values can be duplicated.
- Insertion order is not preserved. and it is based on hashCode of keys.
- Heterogenous objects are allowed for both keys and values.
- Null is not allowed for both key and values, Otherwise we will get NullPointerException.
- Every method present in Hashtable is synchronized and hence, Hashtable object is threadsafe.

- Implement Serializable, Closeable but not RandomAccess
- Hashtable is the best choice if our frequent operation is search operation.

### Constructors:-

Hashtable h = new Hashtable();

- Creates an empty Hashtable object with the default initialCapacity 11 under default fill ratio 0.75.

Hashtable h = new Hashtable(int initialCapacity);

Hashtable h = new Hashtable(int initialCapacity, float fillRatio);

Hashtable h = new Hashtable(Map m);

- if initial capacity of Hashtable is 28 then,

Hashtable h = new Hashtable(28);

then the output is,

{23=E, 16=F, 15=D, 6=C, 5=A, 2=B}

☰

Eg:-

29	
23	23=E
16	16=F
15	15=D
6	6=C
5	5=A
2	2=B
0	

### Properties:-

- It is the child class of Hashtable.
- In the properties both key and Value should be String type.
- In our program if any thing which changes frequently like Username, password etc. are not recommended to hard code in java program. The problem in this approach is for every change we have to recompile, Rebuild and <sup>the appn.</sup> Sometimes even server restart also require which create a big business.

To overcome these problem we should go for properties file. We have to configure such type of variable things in 'properties' file and from java program, we have to read those properties.

If there is any changed in the properties file just Redeployment is enough which won't create business impact to the client.

We can use properties object to hold properties which are coming from properties file.

### Constructors:-

```
properties p = new Properties();
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### Methods:-

① String. setProperty (String name, String value)

→ To set a new Property.

② String getProperty (String name);

→ To get value associated with the specified property.

### ④ Enumeration propertyNames()

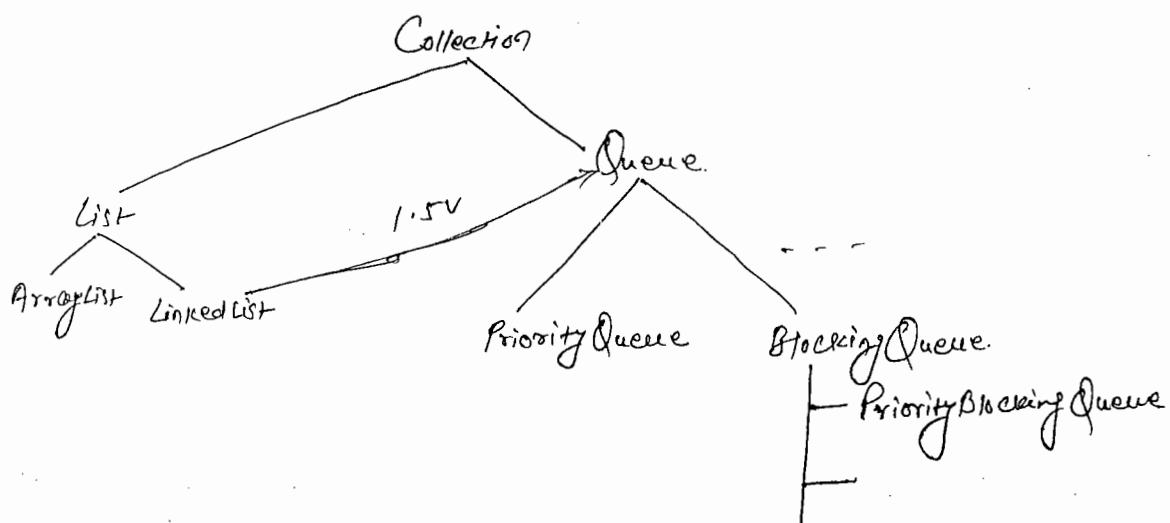
④ void load(InputStream is)

→ to load properties from properties file into java properties object

⑤ void store(OutputStream os, String comment)

→ To store properties from java properties object into properties file.

### 1.5v Enhancements (Queue Interface):-



→ It is the child interface of Collection.

→ If we want to represent a group of individual objects prior to process then we should go for Queue.

~~Before sending sms msg. we have to store all mobile no. in some data~~

for this FIFO requirement, we should go for Queue.

→ Usually Queue follow FIFO order, but based on our requirement we can implement our own priority order also.

- from 1.5v onwards linkedlist class also implement QueueInterface.
- linkedlist based implementation of Queue always follows first in First Out (FIFO) order.

### Queue interface Specific Methods:-

boolean offer(Object o)

- to add an object into the queue.

Object poll()

- to remove and return head element of the queue. If queue element is empty then this method returns null.

Object remove()

- to remove and return head element of the queue. If queue is empty then this method raises RE: NoSuchElementException.

Object peek()

- to return head element of the queue. if queue is empty then this method returns null.

Object element()

- to return head element of the queue. If queue is empty then this method raises RE: NoSuchElementException.

- It is the data structure to store a group of individual objects prior to processing according to some priority.
- The priority can be either default natural sorting order or Customized sorting order specified by Comparator.
- Duplicates objects are not allowed.

(104)

- Insertion Order is not preserved and it is based on some priority.
- If we are depending on default natural sorting order then object should be homogeneous and Comparable. Otherwise we will get ECF.
- If we are defining our own sorting Comparator then the Objects are need not be homogeneous and Comparable.
- Null insertion is not possible. Even as first element also.

### Constructors:-

PriorityQueue q = new PriorityQueue();

- Creates an empty Priority Queue with default initialCapacity and all Objects will be inserted according to default natural sorting order.

PriorityQueue q = new PriorityQueue( int initialCapacity );

( int initialCapacity, Comparator c );

( SortedSet S );

( Collection c );

Note:- Some Operating systems may not provide proper support for Priority Queue.

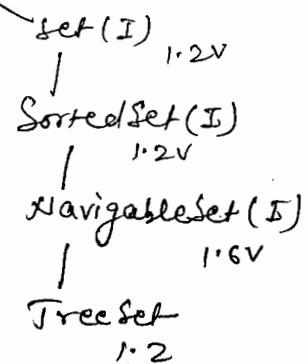


## 1.6v Enhancements

### NavigableSet(I):-

- It is the Child interface of SortedSet.
- It defines several methods for navigation purposes.

### Collection(I) 1.2v



#### ① floor(e)

→ It returns highest element which is  $\leq e$

#### ② lower(e)

→ It returns highest element which is  $< e$

#### ③ ceiling(e)

It returns lowest element which is  $\geq e$ .

#### ④ higher(e)

It returns lowest element which is  $> e$

#### ⑤ pollFirst()

remove and return first element.

#### ⑥ pollLast()

remove and return last element.

#### ⑦ descendingSet()

It returns NavigableSet in reverse order.

(105)

### NavigableMap :-

- It is the child interface of SortedMap.
- It defines several methods for navigation purposes.

floorKey(e)  
lowerKey(e)  
CeilingKey(e)  
higherKey(e)  
pollFirstEntry()  
pollLastEntry()  
descendingMap()

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

# Collection framework

(106)

DURGASOFT



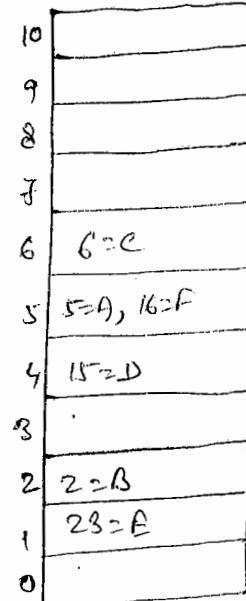
```

1 Program : Demo Program for Hashtable:
2 =====
3 import java.util.*;
4 class HashtableDemo
5 {
6 public static void main(String[] args)
7 {
8 Hashtable h = new Hashtable();
9 h.put(new Temp(5), "A");
10 h.put(new Temp(2), "B");
11 h.put(new Temp(6), "C");
12 h.put(new Temp(15), "D");
13 h.put(new Temp(23), "E");
14 h.put(new Temp(16), "F");
15 //h.put("durga",null); //NPE
16 System.out.println(h);
17 }
18 }
19 class Temp
20 {
21 int i;
22 Temp(int i)
23 {
24 this.i = i;
25 }
26 public int hashCode()
27 {
28 return i;
29 }
30 public String toString()
31 {
32 return i+"";
33 }
34 }
```

Program : Demo Program for Properties:

```

36 =====
37 import java.util.*;
38 import java.io.*;
39 class PropertiesDemo
40 {
41 public static void main(String[] args) throws Exception
42 {
43 Properties p = new Properties();
44 FileInputStream fis = new FileInputStream("abc.properties");
45 p.load(fis);
46 System.out.println(p);
47 String s = p.getProperty("venki");
48 System.out.println(s);
49 p.setProperty("nag","88888");
50 FileOutputStream fos = new FileOutputStream("abc.properties");
51 p.store(fos,"Updated by Durga for SCJP Demo class");
52 }
}
```



O/P:- { 6=C, 16=F, 5=A, 15=D, 2=B, 1=E }

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.



```

53 }
54 abc.properties
55 User=Scott
56 Pwd=tiger
57 Venku=9999.
58
59
60
61
62
63 Program : Demo Program2 for Properties:
64 =====
65 class PropertiesDemo2
66 {
67 public static void main(String[] args)
68 {
69 Properties p = new Properties();
70 FileInputStream fis = new FileInputStream("db.properties");
71 p.load(fis);
72 String url=p.getProperty("url");
73 String user=p.getProperty("user");
74 String pwd=p.getProperty("pwd");
75 Connection con= DriverManager.getConnection(url,user,pwd);
76 ;;;;;
77 }
78 }
79
80
81 Program : Demo Program for PriorityQueue Demo:
82 =====
83 import java.util.*;
84 class PriorityQueueDemo
85 {
86 public static void main(String[] args)
87 {
88 PriorityQueue q = new PriorityQueue();
89 //System.out.println(q.peek()); //null
90 //System.out.println(q.element()); //RE:NSEE
91 for(int i = 0; i<=10 ; i++)
92 {
93 q.offer(i);
94 }
95 System.out.println(q);//[0,1,2.....10]
96 System.out.println(q.poll());//0
97 System.out.println(q);//[1,2,3.....,10]
98 }
99 }
100 Program : Demo Program for PriorityQueue Demo(Customized Priority):
101 =====
102 import java.util.*;
103 class PriorityQueueDemo2
104 {

```

**DURGASOFT**

```

105 public static void main(String[] args)
106 {
107 PriorityQueue q = new PriorityQueue(15,new MyComparator());
108 q.offer("A");
109 q.offer("Z");
110 q.offer("L");
111 q.offer("B");
112 System.out.println(q); // [Z,L,B,A]
113 }
114 }
115 class MyComparator implements Comparator
116 {
117 public int compare(Object obj1, Object obj2)
118 {
119 String s1 = (String) obj1;
120 String s2 = obj2.toString();
121 return s2.compareTo(s1);
122 }
123 }
124
125 Program : Demo Program for NavigableSet Demo:
126 =====
127 import java.util.*;
128 class NavigableSetDemo
129 {
130 public static void main(String[] args)
131 {
132 TreeSet<Integer> t = new TreeSet<Integer>();
133 t.add(1000);
134 t.add(2000);
135 t.add(3000);
136 t.add(4000);
137 t.add(5000);
138 System.out.println(t); // [1000, 2000, 3000, 4000, 5000]
139 System.out.println(t.ceiling(2000)); // 2000
140 System.out.println(t.higher(2000)); // 3000
141 System.out.println(t.floor(3000)); // 3000
142 System.out.println(t.lower(3000)); // 2000
143 System.out.println(t.pollFirst()); // 1000
144 System.out.println(t.pollLast()); // 5000
145 System.out.println(t.descendingSet()); // [4000, 3000, 2000]
146 System.out.println(t); // [2000, 3000, 4000].
147 }
148 }
149
150 Program : Demo Program for NavigableMap Demo:
151 =====
152 import java.util.*;
153 class NavigableMapDemo
154 {
155 public static void main(String[] args)
156 {

```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.



## DURGASOFT

```
157 TreeMap<String, String> t = new TreeMap<String, String>();
158 t.put("b", "banana");
159 t.put("c", "cat");
160 t.put("a", "apple");
161 t.put("d", "dog");
162 t.put("g", "gun");
163 System.out.println(t); // {a=apple, b=banana, c=cat, d=dog, g=gun}
164 System.out.println(t.ceilingKey("c")); // c
165 System.out.println(t.higherKey("e")); // g
166 System.out.println(t.floorKey("e")); // d
167 System.out.println(t.lowerKey("e")); // d
168 System.out.println(t.pollFirstEntry()); // a=apple.
169 System.out.println(t.pollLastEntry()); // g=gun
170 System.out.println(t.descendingMap()); // {d=dog, c=cat, b=banana}
171 System.out.println(t); // {b=banana, c=cat, d=dog}
172 }
173 }
174 }
175
176 Program -1: To sort elements of the List based on Natural Sorting Order:
177 ======
178 import java.util.*;
179 class CollectionsSortDemo
180 {
181 public static void main(String[] args)
182 {
183 ArrayList l = new ArrayList();
184 l.add("Z");
185 l.add("A");
186 l.add("K");
187 l.add("N");
188 // l.add(new Integer(10));---CCE
189 l.add(null);---NPE
190 System.out.println("Before Sorting: " + l);
191 Collections.sort(l);
192 System.out.println("After Sorting: " + l);
193 }
194 }
195 Program -2: To sort elements of the List based on Customized Sorting Order:
196 ======
197 import java.util.*;
198 class CollectionsSortDemo2
199 {
200 public static void main(String[] args)
201 {
202 ArrayList l = new ArrayList();
203 l.add("Z");
204 l.add("A");
205 l.add("K");
206 l.add("L");
207 System.out.println("Before Sorting: " + l);
208 Collections.sort(l, new MyComparator());
209 }
```



**DURGASOFT**

```

209 System.out.println("After Sorting:" +l);
210
211 }
212 }
213 class MyComparator implements Comparator
214 {
215 public int compare(Object obj1, Object obj2)
216 {
217 String s1 = (String) obj1;
218 String s2 = obj2.toString();
219 return s2.compareTo(s1);
220 }
221 }
222 Program -3: To search elements of the List
223 =====
224 import java.util.*;
225 class CollectionsSearchDemo
226 {
227 public static void main(String[] args)
228 {
229 ArrayList l = new ArrayList();
230 l.add("Z");
231 l.add("A");
232 l.add("M");
233 l.add("K");
234 l.add("a");
235 System.out.println(l);
236 Collections.sort(l);
237 System.out.println(l);
238 System.out.println(Collections.binarySearch(l, "Z"));
239 System.out.println(Collections.binarySearch(l, "J"));
240 }
241 }
242 Program -4: To search elements of the List
243 =====
244 import java.util.*;
245 class CollectionsSearchDemo1
246 {
247 public static void main(String[] args)
248 {
249 ArrayList l = new ArrayList();
250 l.add(15);
251 l.add(0);
252 l.add(20);
253 l.add(10);
254 l.add(5);
255 System.out.println(l);
256 Collections.sort(l, new MyComparator());
257 System.out.println(l);
258 System.out.println(Collections.binarySearch(l, 10, new MyComparator())); //2
259 System.out.println(Collections.binarySearch(l, 13, new MyComparator())); //3
260 System.out.println(Collections.binarySearch(l, 17)); // unpredictable

```



**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

```
261
262 }
263 }
264
265 class MyComparator implements Comparator
266 {
267 public int compare(Object obj1, Object obj2)
268 {
269 Integer i1 = (Integer)obj1;
270 Integer i2 = (Integer)obj2;
271 return i2.compareTo(i1);
272 }
273 }
274
275 Program -5: To reverse elements of the List
276 =====
277 import java.util.*;
278 class CollectionsReverseDemo
279 {
280 public static void main(String[] args)
281 {
282 ArrayList l = new ArrayList();
283 l.add(15);
284 l.add(0);
285 l.add(20);
286 l.add(10);
287 l.add(5);
288 System.out.println(l);
289 Collections.reverse(l);
290 System.out.println(l);
291 }
292 }
293
294 Program -6: To sort elements of the Array
295 =====
296 import java.util.Arrays;
297 import java.util.Comparator;
298 class ArraysSortDemo
299 {
300 public static void main(String[] args)
301 {
302 int[] a = {10,5,20,11,6};
303 System.out.println("Primitive Array before sorting:");
304 for(int a1 : a)
305 {
306 System.out.println(a1);
307 }
308 Arrays.sort(a);
309 System.out.println("Primitive Array After sorting:");
310 for(int a1 : a)
311 {
312 System.out.println(a1);}
```



```

313 }
314 String[] s = {"A","Z","B"};
315 System.out.println("Object Array Before sorting:");
316 for(String a2 : s)
317 {
318 System.out.println(a2);
319 }
320 Arrays.sort(s);
321 System.out.println("Object Array After sorting:");
322 for(String a1 : s)
323 {
324 System.out.println(a1);
325 }
326 Arrays.sort(s, new MyComparator());
327 System.out.println("Object Array After sorting by comparator:");
328 for(String a1 : s)
329 {
330 System.out.println(a1);
331 }
332 }
333 }
334 }
335 class MyComparator implements Comparator
336 {
337 public int compare(Object o1, Object o2)
338 {
339 String s1 = o1.toString();
340 String s2 = o2.toString();
341 return s2.compareTo(s1);
342 }
343 }
344 Program -7: To search elements of the Array
345 =====
346 import java.util.*;
347 import static java.util.Arrays.*;
348 class ArraysSearchDemo
349 {
350 public static void main(String[] args)
351 {
352 int[] a = {10,5,20,11,6};
353 Arrays.sort(a); //sort by natural order
354 System.out.println(Arrays.binarySearch(a,6)); //1
355 System.out.println(Arrays.binarySearch(a,14)); //-5
356
357 String[] s = {"A","Z","B"};
358 Arrays.sort(s);
359 System.out.println(binarySearch(s,"Z")); //2
360 System.out.println(binarySearch(s,"S")); //-3
361
362 Arrays.sort(s, new MyComparator());
363 System.out.println(binarySearch(s,"Z", new MyComparator())); //0
364 System.out.println(binarySearch(s,"S", new MyComparator())); //-2

```



## DURGASOFT

```
365 System.out.println(binarySearch(s,"N")); //unpredictable result
366
367 }
368 }
369
370 class MyComparator implements Comparator
371 {
372 public int compare(Object o1, Object o2)
373 {
374 String s1 = o1.toString();
375 String s2 = o2.toString();
376 return s2.compareTo(s1);
377 }
378 }
379 Program -8: To convert Array to List
380 =====
381 import java.util.*;
382 class ArraysAsListDemo
383 {
384 public static void main(String[] args)
385 {
386 String[] s = {"A","Z","B"};
387 List l = Arrays.asList(s);
388 System.out.println(l); // [A,Z,B]
389 s[0] = "K";
390 System.out.println(l); // [K,Z,B]
391 l.set(1,"L");
392 for(String s1 : s)
393 System.out.println(s1); // K,L,B
394 // l.add("durga"); // USOE
395 // l.remove(2); // USOE
396 l.set(1,new Integer(10)); // ASE
397
398 }
399 }
400
401 Collections FAQs
402 =====
403 Q1. What are limitations of object Arrays?
404 Q2. What are differences between arrays and collections?
405 Q3. what are differences between arrays and ArrayList?
406 Q4. What are differences between arrays and Vector?
407 Q5. What is Collection API ?
408 Q6. What is Collection framework?
409 Q7. What is difference between Collections and Collection?
410 Q8. Explain about Collection interface?
411 Q9. Explain about List interface?
412 Q10. Explain about Set interface?
413 Q11. Explain about SortedSet interface?
414 Q12. Explain about NavigableSet ?
415 Q13. Explain about Queue interface?
416 Q14. Explain about Map Interface?
```



(16)

**DURGASOFT**

417. Q15. Explain about SortedMap ?  
418. Q16. Explain about NavigableMap?  
419. Q17. Explain about ArrayList class?  
420. Q18. What Is RandomAccess Interface?  
421. Q19. Explain about LinkedList class?  
422. Q20. Explain about Vector class?  
423. Q21. What is difference between ArrayList and Vector?  
424. Q22. How we can get synchronized version of ArrayList?  
425. Q23. What is difference between size and capacity of a Collection Object?  
426. Q24. What is difference between ArrayList and Linked List?  
427. Q25. What are legacy classes and interfaces present in Collections framework ?  
428. Q26. what is difference Enumeration and Iterator?  
429. Q27. What are limitations of Enumeration?  
430. Q28. What is difference between enum and Enumeration?  
431. Q29. What Is difference between Iterator and ListIterator?  
432. Q30. What is relation between ListIterator and Iterator?  
433. Q31. Explain about HashSet class?  
434. Q32. If we are trying to insert duplicate values in Set what will happen?  
435. Q33. What Is LinkedHashSet?  
436. Q34. Differences between HashSet and LinkedHashSet?  
437. Q35. What are major enhancements in 1.4 version of collection frame work?  
438. Q36. Explain about TreeSet?  
439. Q37. What are differences between List and Set interfaces?  
440. Q38. What is Comparable interface?  
441. Q39. What is Comparator interface?  
442. Q40. What are differences between Comparable and Comparator?  
443. Q41. What is difference between HashSet and TreeSet?  
444. Q42. What is Entry interface?  
445. Q43. Explain about HashMap?  
446. Q44. Explain about LinkedHashMap?  
447. Q45. Differences between HashMap and LinkedHashMap ?  
448. Q46. Differences between HashMap and Hashtable?  
449. Q47. What is IdentityHashMap?  
450. Q48. What is difference between HashMap and IdentityHashMap?  
451. Q49. What is WeakHashMap?  
452. Q50. What is difference between HashMap and WeakHashMap?  
453. Q51. What is TreeMap?  
454. Q52. What is Hashtable  
455. Q53. What is PriorityQueue?  
456. Q54. What is Arrays class?  
457. Q55. We are planning to do an indexed search in a list of objects. Which of the two Java collections should you use: ArrayList or LinkedList?  
458. Q56. Why ArrayList is faster than Vector?



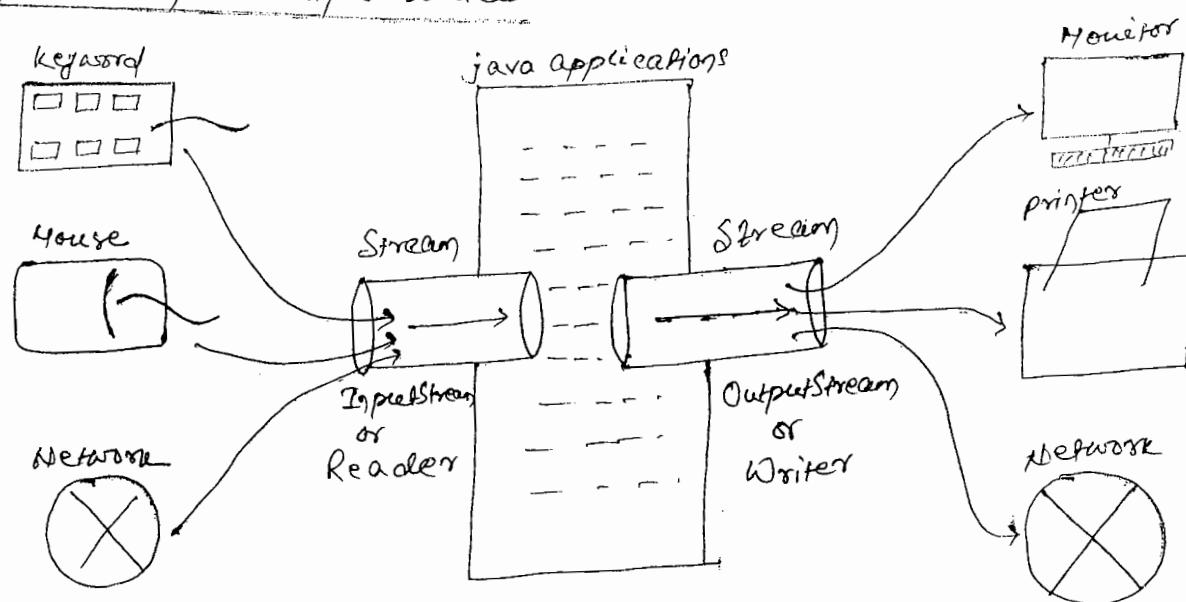


## I/O Streams:-

As part of C and C++ applications, to perform I/O operations we have to use some predefined library like `printf()`, `scanf()`, `cin>>`, `cout<<`, ...

Similarly, in java applications, to perform I/O operations we have to use "Streams".

Stream is a channel or medium, it able to allow data in continuous flow from input device to java program (appn) and from java application to output device.



There are two types of Streams in java

- (1) Byte Oriented Stream
- (2) Character Oriented Stream.

### (1) Byte Oriented Stream:-

These streams are able to allow data in continuous flow in the form of bytes from input device to java program and from java program to output.

There are two types of Byte Oriented Streams.

- (1) Input Streams
- (2) Output Streams.

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## ① Input Streams:-

These are byte oriented streams, which are able to allow data in continuous flow in the form of bytes from input devices to java programs.

Eg:-

InputStream ✓  
ByteArrayInputStream ✓  
FileInputStream ✓  
BufferedInputStream ✓  
DataInputStream ✓  
ObjectInputStream ✓

— — — — —  
— — — — —  
— — — — —

## ② Output Stream:-

These are byte Oriented Streams, which are able to allow data in continuous flow in the form of bytes from java programs to output devices.

Eg:-

OutputStream ✓  
ByteArrayOutputStream ✓  
FileOutputStream ✓  
BufferedOutputStream ✓  
DataOutputStream ✓  
ObjectOutputStream ✓

— — — — —  
— — — — —

Note:- In byte Oriented Streams, length of each and every data item is 1 byte.

Note:- All the byte Oriented Stream classes are ended with "Stream".

(2)  
118

## ② Character Oriented Streams:-

These Streams are able to allow data in continuous form in the form of characters from input devices to java programs and from java programs to output devices.

There are two types of Character Oriented Streams.

① Reader

② Writer.

### ① Reader :-

It is a character Oriented Stream, it able to <sup>allow</sup> data in continuous flow in the form of characters from input devices to java programs.

Ex:- Reader

CharArrayReader

FileReader

BufferedReader

InputStreamReader

StringBufferedReader

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

### ② Writer :-

It is a character Oriented Stream, it able to allow data in continuous flow in the form of characters from java appn to output device.

Ex:- Writer

FileWriter

BufferedWriter

FilterWriter

StringBufferedWriter

PrintWriter

[Note:- In character Oriented Streams, length of each and every data item is 2 bytes.]

[Note:- In the character Oriented Streams all the classes are ended with either reader or writer.]

## File Operations:-

① In java applications, it is convention to transfer the data from a particular source file to java application and from java application to a particular target file.

+ f.

To perform file Operations, JAVA has provided the following predefined classes.

- ① FileOutputStream
- ② FileInputStream
- ③ FileWriter
- ④ FileReader.

### ① FileOutputStream :-

The main purpose of FileOutputStream is to carry data in the form of bytes from java application to a particular target file.

To perform the above output operation by using FileOutputStream we have to use the following steps.

#### ① Create FileOutputStream class Object:

To Create FileOutputStream class object, we have to use the following Constructors.

`public FileOutputStream (String file-Name)`

→ It will Override old data with new data at each and every write operation.

`public FileOutputStream (String file-Name, boolean b)`

→ If 'b' value is true then FileOutputStream will not override old data with new data, new data will be appended to old data.

(3)

(19)

Ex:-

```
FileOutputStream fos = new FileOutputStream("abc.txt", true);
```

When JVM encounter the above instruction, JVM will perform the following actions.

- (1) JVM will take the specified file name.
- (2) JVM will check whether the specified file is existed or not at the specified location.
- (3) If the specified file is not existed then JVM will create a new file with the same name.
- (4) If the specified file is existed then JVM will create FileOutputStream.

## (2) Declare data and Convert into byte[]:-

```
String data = "Hello";
byte[] b = data.getBytes();
```

To convert data from String type to byte[] we have to use the following method from java.lang.String class.

```
public byte[] getBytes()
```

## (3) Write byte[] data to FileOutputStream:-

To write byte[] data to FileOutputStream we have to use the following method.

```
java.io.FileOutputStream
```

```
public void write(byte[] b)
```

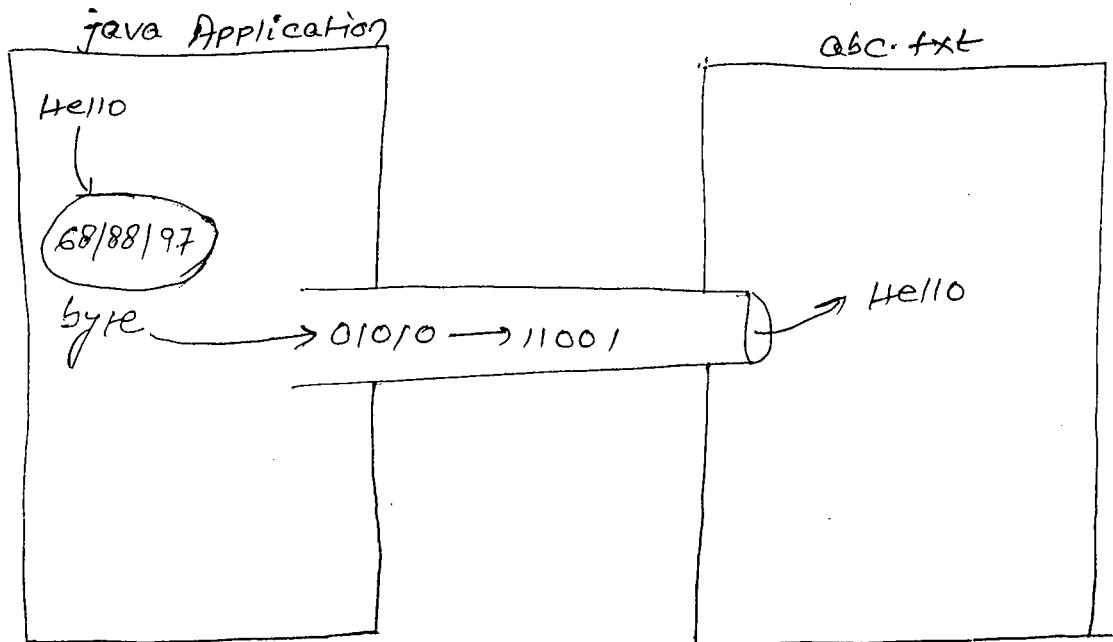
```
Ex:- fos.write(b)
```

## (4) Close FileOutputStream:-

To close FileOutputStream, we have to use the following method.

```
public void close() throws IOException
```

```
Ex:- fos.close();
```



`FileOutputStream fos = new FileOutputStream("abc.txt", true)`

Ex:-

```

import java.io.*;
public class FileOutputStreamEx
{
 public void main(String [] args) throws Exception
 {
 FileOutputStream fos = new FileOutputStream("C:/abc/
xyz/abc.txt");
 String data = "Durga Software Solution";
 byte[] b = data.getBytes();
 fos.write(b);
 fos.close();
 }
}

```

## FileInputStream:-

This byte oriented stream can be used to carry data from a particular source file to java program in the form of byte. To perform this input operation by using FileInputStream then we have to use the following step.

### ① Create FileInputStream Object:-

To create FileInputStream Class object we have to use the following constructor from

java.io.FileInputStream Class  
public FileInputStream(String file-name)

Ex:-

FileInputStream f1 = new FileInputStream("abc.txt")

where jvm will encounter the above instruction jvm will perform the following action.

- ① JVM will take specific file name.
- ② JVM will check specific file name at specific location.
- ③ If specific file is not available at specific location then it rise an exception like java.io.FileNotFoundException.
- ④ If the Specific file is existed at the Specific location then JVM will create FileInputStream from source file to java application.
- ⑤ When FileInputStream is created then the complete data which is available in the source file is transmitted to FileInputStream in the form of byte representation.

## ② Get Data size from FileInputStream and prepare byte[] with the data size:-

To get data size from FileInputStream we have to use the following method from java.io.FileInputStream.

`public int available()` -

Ex:- `int size = fis.available();` ✓  
`byte[] b = new byte[size];` ✓

## ③ Read data from FileInputStream into byte[] :-

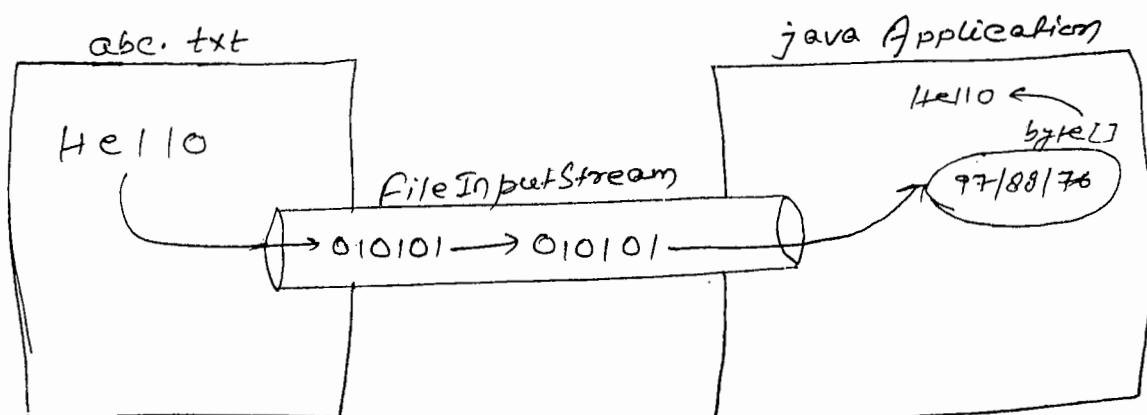
To read data from FileInputStream, we have to use the following method from java.io.FileInputStream.

`public void read(byte[] b)` -

Ex:- `fin.read(b);` ✓  
`String data = new String(b);` ✓  
`S.O.P(data);` ✓

## ④ Close FileInputStream :-

`fin.close();` ✓



(12) 5

Q Display the content of your file on command prompt by taking file name as command line input.

```
→ import java.io.*;
Class fileContentEx
{
 public void main(String[] args) throws Exception
 {
 String file_name = args[0];
 FileInputStream fis = new FileInputStream(file_name);
 int size = fis.available();
 byte[] b = new byte[size];
 fis.read(b);
 String data = new String(b);
 System.out.println(data);
 fis.close();
 }
}
```

Q Count the no. of words which are available in a particular text file [file name: abc.txt].

```
→ import java.io.*;
Class WordCountEx
{
 public void main(String[] args) throws Exception
 {
 FileInputStream fin = new FileInputStream("abc.txt");
 byte[] b = new byte[size];
 fis.read(b);
 String data = new String(b);
 String[] str = data.split(" ");
 }
}
```

```

S.o.println("No. of words :" + str.length);
int Count = 0;
for (String s : str)
{
 if (s.equals("Durga"))
 {
 Count = Count + 1;
 }
}
s.o.println("Durga is repeated :" + Count);
fis.close();
}

```

Q. Write a java program to copy an image from a particular source file to a particular target file.

Source file Name:- Desert.jpg

Target file Name:- Desert-Desert.jpg

```

→ import java.io.*;
class ImgCopyEx
{
 public void main (String [] args) throws Exception
 {
 FileInputStream fin = new FileInputStream ("Desert.jpg");
 int size = fin.available ();
 byte [] b = new byte [size];
 fis.read ();
 -- -- -- if os = new -- -- -- -
 fos.write (b);
 fin.close ();
 fos.close ();
 }
}

```

## File Reader:-

(7)  
124

This character Oriented Stream can be used to transfer the data from a particular source file to java application.

To perform this input Operation by using FileReader, we have to use the following steps.

### ① Create FileReader Class Object:-

To Create FileReader class object, we have to use the following constructor from java.io.FileReader.

```
public FileReader(String file-name) throws IOException
```

Ex:-

```
FileReader fr = new FileReader("abc.txt");
```

→ When JVM encounter the above instruction JVM will perform the following action

- ① JVM will take the Specified source file name.
- ② JVM will check whether Specified file is exist or not at Specified file location.
- ③ If the Specified file is not available then jvm will rise an exception like java.io.FileNotFoundException.
- ④ If the specified file is existed, then jvm will create a FileReader from source file to java Application.
- ⑤ When FileReader is created then complete data which is available in source file, transferred to FileReader in the form of character.

≡

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

## ② Read data from FileReader:-

- (a) Read character by character in the form of ASCII value from FileReader.
- (b) Convert all the ASCII value into respective character.
- (c) Concat all the characters to a String variable.
- (d) Repeat above process upto the "end of file" character that is '-1'.

Ex:-

```
String data = " ";
int val = fr.read();
while(val != -1)
{
 data = data + (char)val;
 val = fr.read();
}
```

To read a single character in the form of ASCII value from FileReader we have to use the following method.

**public int read()**

## ③ Close FileReader:-

```
fr.close();
```

Ex:-

```
import java.io.*;
class FileReaderex
{
 public static void main(String [] args) throws Exception
 {
 FileReader fr = new FileReader("abc.txt");
 String data = " ";
 int val = fr.read();
 while(val != -1)
```

8  
125

```

data = data + (char)val;
val = fr.read();
}
s.o.println(data);
fr.close();
}
}
}

```

### FileWriter:-

This character Oriented Stream can be used to carry data from java application to a particular target file.

To perform the above output operation by using FileWriter then we have to use the following steps.

#### ① Create FileWriter class object:-

To create FileWriter class object we have to use the following Constructors from java.io.FileWriter class.

public FileWriter(String file-name).

→ It will override old data with new data at each and every write operation.

public FileWriter(String file-name, boolean b)

→ It will append new data to old data existed in the file, at each and every file operation if 'b' value is true.

Ex:-

FileWriter fw = new FileWriter("abc.txt", true);

#### Internal flow:-

① JVM will take the specified file name and search for it at the specified location.

- (2) If file is not existed then Jav will create a new file with the same name.
- (3) If the file existed then Jav will create FileWriter.

## (2) Declare the data and convert that data into Char[]:-

→ To convert String data into Char[], we have to use the following method from java.lang.String class.

public char[] toCharArray()

Ex:- String data = "Hello";

Char[] ch = data.toCharArray();

## (3) Write Char[] data to FileWriter:-

To write Char[] data to FileWriter we have to use the following method.

public void write(Char[] ch)

Ex:- fw.write(ch);

## (4) Close FileWriter:-

fw.close();

Ex:- import java.io.\*;

{  
class FileWriterEx

}  
public (String[] args) throws Exception

FileWriter fw = new FileWriter("xyz.txt", true);

String data = "Durga Software Solutions";

Char[] ch = data.toCharArray();

fw.write(ch);

fw.close();

}

(9)

(126)

## Serialization and Deserialization :-

If we design any java application on the basis of client-server arch. then that application is called as distributed application.

→ In distributed applications, it is convention to transfer an object from local machine to remote machine.

To transfer an object from local machine to remote machine we have to use the following steps.

### At Local Machine:-

- (a) Create an object which we want to transfer.
- (b) Separate data from an object which we want to transfer.
- (c) Convert data from System representation to Network representation
- (d) Send data to network.

### At Remote Machine:-

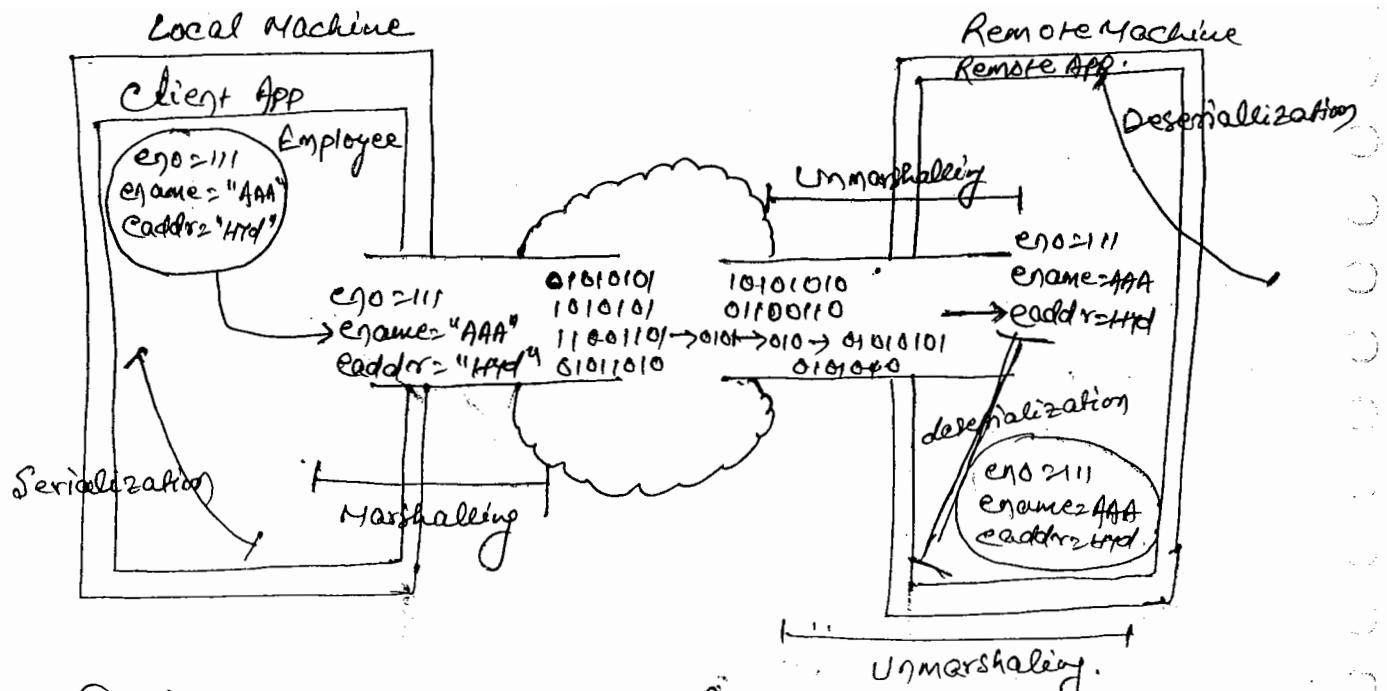
- (a) Get data from network, convert data from network representation to S/m representation.
- (b) Re-Construct an object on the basis of data.

The process of converting data from S/m representation to network representation is called as "Marshalling".

The process of converting data from <sup>network</sup> representation to S/m representation is called as Unmarshalling.

The process of separating data from an object is called as "Serialization".

The process of reconstructing an object on the basis of data is called as Deserialization".



In java application to perform serialization and deserialization, java has provided the following two byte-Oriented Streams.

- ① "ObjectOutputStream" for serialization.
- ② "ObjectInputStream" for deserialization.

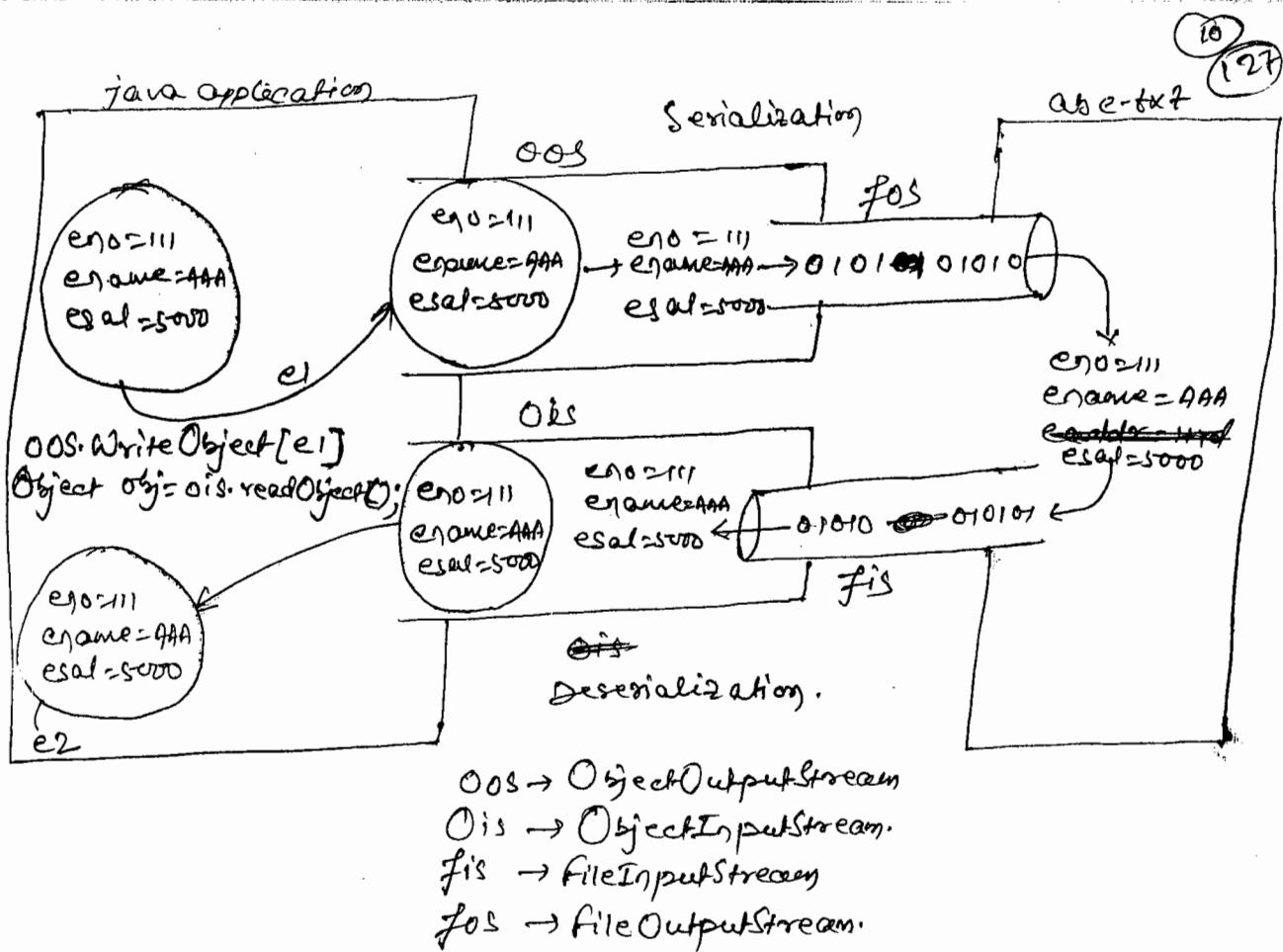
In java applications to perform serialization we have to use FileOutputStream along with ObjectOutputStream and to perform deserialization we have to use FileInputStream along with ObjectInputStream.

In Standalone applications, if we want to perform serialization and deserialization then we have to use a file to manage serialized data.

#### Steps to perform Serialization:-

- ① Create Serializable Object.

In java appn all the objects are not eligible for serialization, Only few objects which are implementing



java.io.Serializable interface are eligible of Serialization and Deserialization.

java.io.Serializable interface is a marker interface, it is not having abstract methods but it will make any object eligible for serialization and deserialization.

Ex:-

Class Employee implements java.io.Serializable.

```

{
 -
 -
}

```

Employee e1 = new Employee();

~~~~~

## ② Create FileOutputStream with the target file:-

FileOutputStream fos = new FileOutputStream("abc.txt");

## ③ Create ObjectOutputStream:-

To Create ObjectOutputStream class we have to use the following constructor from java.io.ObjectOutputStream class.

public ObjectOutputStream (FileOutputStream fos)

Ex:-

ObjectOutputStream oos = new ObjectOutputStream(fos);

## ④ Write Serializable Object to ObjectOutputStream:-

To Write Serializable Object to ObjectOutputStream we have to use the following method.

Public void writeObject (Object Obj) throws NotSerializableException.

Ex:- oos.writeObject(e1)

Where if 'e1' referred class is not implementing Serializable interface then JVM will raise an exception like java.io.NotSerializableException.

Note:- In the above process when we give Serializable Object to ObjectOutputStream will perform the required serialization and it will send the generated serialized data to FileOutputStream, where FileOutputStream will send that serialized data to target file.

3

(11)  
(12)

## Steps to perform Deserialization:-

### ① Create FileInputStream Object :-

FileInputStream fis = new FileInputStream("abc.txt");

When we create FileInputStream, automatically serialized data will be transferred to FileInputStream.

### ② Create ObjectInputStream class Object :-

To create ObjectInputStream class Object we have to use the following constructor.

public ObjectInputStream(FileInputStream fis)

For -

ObjectInputStream ois = new ObjectInputStream(fis);

When we create ObjectInputStream, automatically serialized data will be send to ObjectInputStream from FileInputStream and ois will perform the required deserialization and it will prepare an object in the form of java.lang.Object type.

### ③ Read Deserialized object from Ois (ObjectInputStream) :-

To read Deserialized Object from Ois we have to use the following method.

public Object readObject()

Employee e2 = (Employee) ois.readObject();

✓

Ex:-

```
import java.io.*;
Class Employee implements Serializable.
{
 int eno;
 String ename;
 float esal;
 String eaddr;
 Employee(int eno, String ename, float esal, String eaddr)
 {
 this.eno = eno;
 this.ename = ename;
 this.esal = esal;
 this.eaddr = eaddr;
 }
 public void getEmployeeDetails()
 {
 System.out.println("Employee Details");
 System.out.println("-----");
 System.out.println("Employee Number :" + eno);
 System.out.println("Employee Salary :" + esal);
 System.out.println("Employee Address :" + eaddr);
 }
}
Class Serialization.
{
 public static void main(String[] args) throws Exception
 {
 FileOutputStream fos = new FileOutputStream("emp.txt");
FileOutputStream("emp.ser")
 ObjectOutputStream oos = new ObjectOutputStream(fos);
 Employee e = new Employee(111, "AAA", 5000, "Hyd");
 oos.writeObject(e);
 }
}
```

```

S.O.P19 ("Employee Details Before Serialization");
e1.getEmployeeDetails();
oos.writeObject(e1);
S.O.P19 ();
FileInputStream fis = new FileInputStream("emp.txt");
Ois ois = new ObjectInputStream(fis);
Employee e2 = (Employee) ois.readObject();
S.O.P19 ("Employee Details after serialization");
e2.getEmployeeDetails();

```

(12)  
(129)

*ObjectInputStream* → must write  
not *fis*. here.

Note:- If we perform serialization over an object with out implementing `java.io.Serializable` interface then JVM will raise an exception like.

"`java.io.NotSerializableException`".

In java applications, if we implement `Serializable` interface at super class then automatically ~~not~~ all the sub class objects are eligible for serialization.

Ex:-

```

import java.io.*;
class Employee implements Serializable
{
 int eno=111;
 String cname="AAA";
}
class Manager extends Employee
{
 String mgr_Email="amarendra.kumar999@gmail.com";
 float mgr_Salaryr=25000.0f;
}

```

Class Test

```
{
 public static void main(String[] args) throws Exception {
 FileOutputStream fos = new FileOutputStream("aaa.txt");
 ObjectOutputStream oos = new ObjectOutputStream(fos);
 Manager m = new Manager();
 oos.writeObject(m);
 }
}
```

→ In java applications, while performing serialization over an object if any associated class object is identified then JVM will perform serialization over associated object also whenever the associated object class is implementing java.io.Serializable interface. In this ~~if~~ context, if the associated object class is not implementing java.io.Serializable interface then JVM will raise an exception like java.io.NotSerializableException.

Ex:- import java.io.\*;

```
class Employee implements Serializable {
 String bid = "B-111";
 String bname = "Axis Bank";
}
```

```
class Account implements Serializable {
```

```
 String accNo = "abc123";
 String accName = "AAA";
 Bank b = new Bank();
}
```

Class Employee implements Serializable.

```
class Employee implements Serializable
{
 int id=111;
 String name = "AAA";
 Account acc = new Account();
}
```

Class Test

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

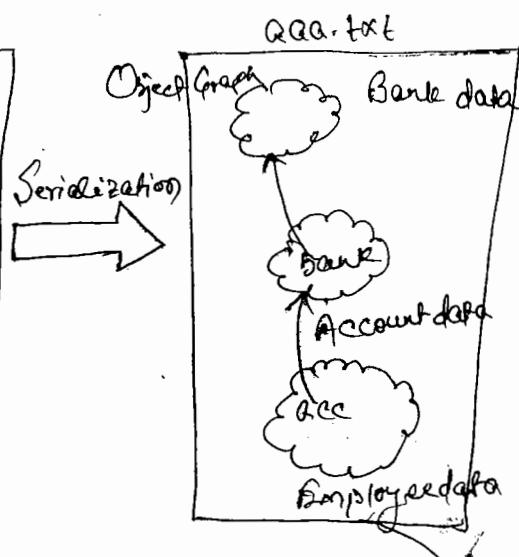
```
 FileOutputStream fos = new FileOutputStream("AAA.txt");
 ObjectOutputStream oos = new ObjectOutputStream(fos);
 Employee emp = new Employee();
 oos.writeObject(emp);
 }
}
```

In java applications, when we perform serialization over associated objects then the object data will be stored in the target file with the same association relation in the form of a graph called as "Object Graph".

```
Class Bank implements Serializable
{
}

Class Account implements Serializable
{
 Bank b = new Bank();
}

Class Employee implements Serializable
{
 Account acc = new Account();
}
```



Note:- In java applications Only instance variables are allowed in serialization, static variables and local variables are not allowed in serialization, because, static variables and local variables are not stored in Object.

Note:- If we don't want to allow any variable in a serialization, then we have to declare that variable as "transient" variable.

Eg:-

```
transient String upwd;
```

Note:- transient variables are getting more security when compared with the variables which are allowed in serialization].

### Externalization:-

As part of serialization and deserialization, developer is not performing serialization directly, developer is giving serialization object to ObjectOutputStream, where ObjectOutputStream will perform the actual serialization, therefore, developer is not having controlling over serialization process.

In java apps, Deserialization is performed by ObjectInputStream directly, where developer will read the deserialized object, therefore developer is not having controlling over deserialization process.

In java apps, if we want to have controlling over serialization and deserialization in order to perform encoding and decoding, encryption and decryption, ... pre-processing and post-processing actions over the data then we have to use "Externalization".

→ If we want to implement Externalization in java application then we have to use the following steps.

P.T.O

## ① Create Externalizable Object:-

(a) Declare an user defined class.

(b) Implement java.io.Externalizable interface.

[Note:- Externalizable interface is a child interface to Serializable interface.]

(c) Implement the following methods of Externalizable interface in user defined class.

① public void writeExternal(ObjectOutput oop) throws IOException

→ It will be executed just before performing serialization

→ Manipulate the data and send manipulated data to Object-Output by using the following methods.

    public void writeXXX(XXX values)

    Where XXX may be byte, int, UTF[String]....

② Public void readExternal(ObjectInput Oip) throws IOException,

ClassNotFoundException

→ This method will be executed immediately after Deserialization.

→ Get Deserialized data from ObjectInput by using the following methods, manipulate data and send that manipulated data to java application.

    public XXX readyXXX()

    Where XXX may be byte, int, UTF[String].....

③ Declare a public and 0-arg Constructor in user defined class.



SRI RAGHAVENDRA XEROX

Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

P.T.O.



② Perform Serialization and Deserialization over Externalization Object:-

This step is same as Serialization and Deserialization. Deserialization steps which we performed in Serialization and Deserialization topic. (for fig. see on the last page).

15-

132

```
import java.io.*;
class Student implements Externalizable
{
 String Sid;
 String Sname;
 String Semail;
 String Smobile;
 Student (String sid, String sname, String semail, String smobile)
 {
 this.Sid = sid;
 this.Sname = sname;
 this.Semail = semail;
 this.Smobile = smobile;
 }
 public Student()
 {
 }
 public void writeExternal (ObjectOutput oop) throws IOException
 {
 try
 {
 String [] Str1 = Sid.split("-");
 int sno = Integer.parseInt (Str1[0]);
 String [] Str2 = Semail.split("@");
 String mail_Id = Str2[0];
 String [] Str3 = Smobile.split("-");
 long mnno = Long.parseLong (Str3[1]);
 oop.writeInt (sno);
 oop.writeUTF (Sname);
 oop.writeUTF (mail_Id);
 oop.writeLong (mnno);
 }
 catch (Exception e)
 {
 e.printStackTrace();
 }
 }
}
```

public void readExternal(ObjectInput oip) throws IOException,  
ClassNotFoundException.

```
{
 Sid = "DSS-" + oip.readInt();
 Sname = oip.readUTF();
 Semail = oip.readUTF() + "@durgasoft.com";
 Smobile = "91-" + oip.readLong();
}
```

public void getStudentDetail()

```
{
 System.out.println("Student Detail");
 System.out.println("-----");
 System.out.println("Student Id : " + sid);
 System.out.println("Student Name : " + sname);
 System.out.println("Student Email : " + semail);
 System.out.println("Student Mobile : " + smobile);
}
```

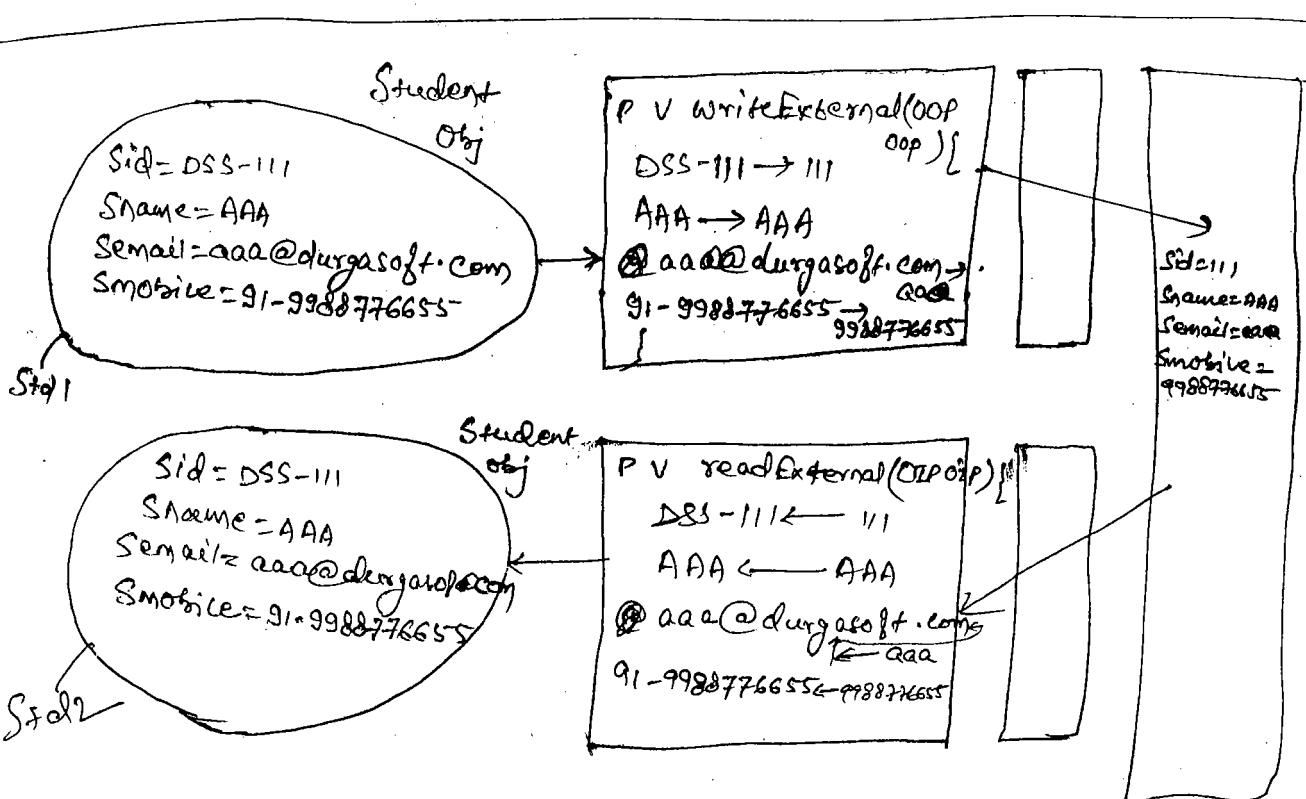
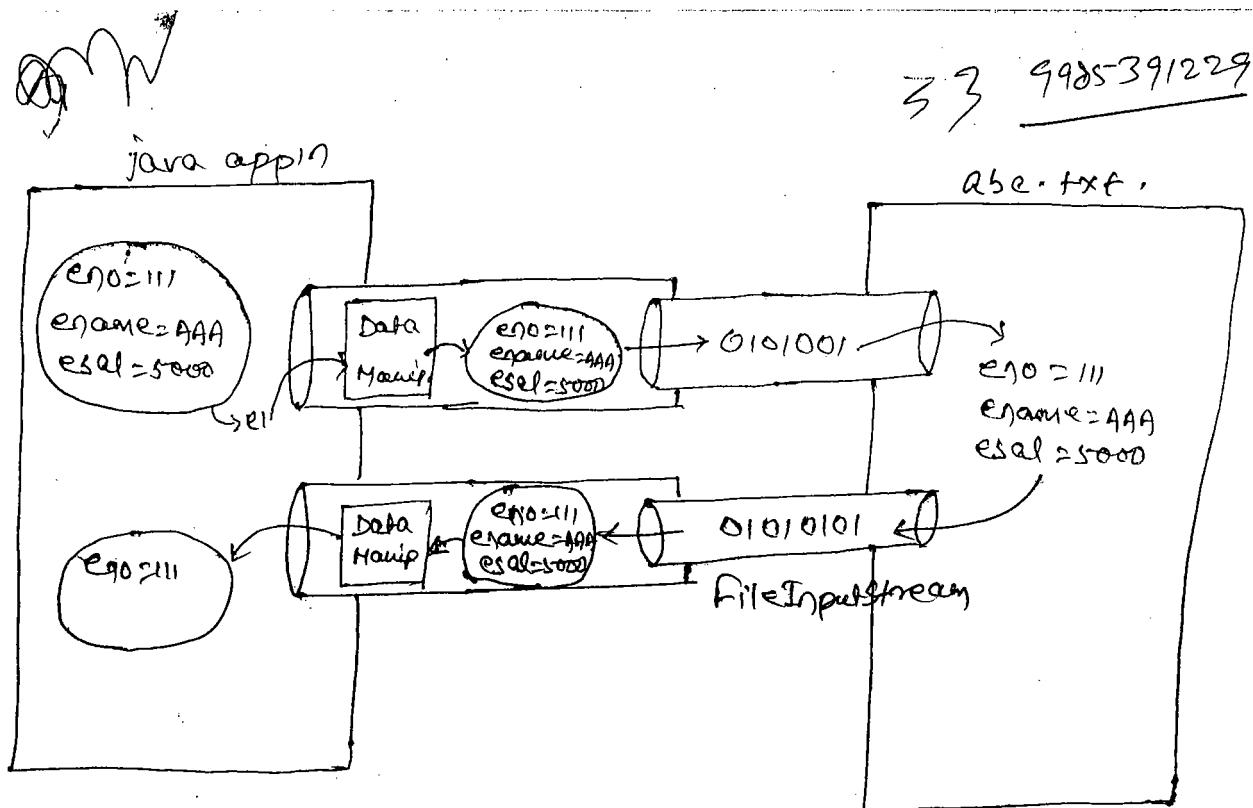
Class Externalization Ex

psvm(String[] args) throws Exception

```
{
 FileOutputStream fos = new FileOutputStream("student.txt");
 ObjectOutputStream oos = new ObjectOutputStream(fos);
 ObjectOutputStream oos = new ObjectOutputStream(fos);
 Student std1 = new Student("DSS-111", "AAA", "aaa@durgasoftware.
 com", "91-9988776655");
 System.out.println("Student details before serialization");
 std1.getStudentDetail();
 oos.writeObject(std1);
 FileInputStream fis = new FileInputStream("student.txt");
 ObjectInputStream
```

```
ObjectInputStream ois = new ObjectInputStream(fis);
Student std2 = (Student) ois.readObject();
System.out.println("Student Details after Deserialization");
std2.getStudentDetail();
} → check here program will finished or not.
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.



7:00am to 9:30am → Annotations (Saturday)

7: am to 8:30 am → TDBC / 8:30 am

8:30am to 1:00pm → JV + Arch. [7 am, 6 pm]

Sunday,

## Dynamic Input Approaches in Java:-

- ① BufferedReader
- ② Scanner
- ③ Console

### 1) BufferedReader:-

If we want to take dynamic input by using BufferedReader approach then we have to use the following instruction.

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

- Where "in" is a static variable defining java.lang.System class, it able to refer a predefined input Stream object which is connected with command prompt.
- If we provide input data on command prompt, then that data will be send at to InputStream object in the form of byte representations.
- Where the purpose of InputStreamReader is to convert data from byte representation to character representation.
- When data is coming to InputStreamReader, we can read data, but it will reduce performance of the input Operation. In this context, to improve the performance of input Operation we have to use BufferedReader, Where BufferedReader is able to store data in the form of buffers.

To read data from BufferedReader we have to use the following methods.

- ① `readLine()`
  - ② `read()`

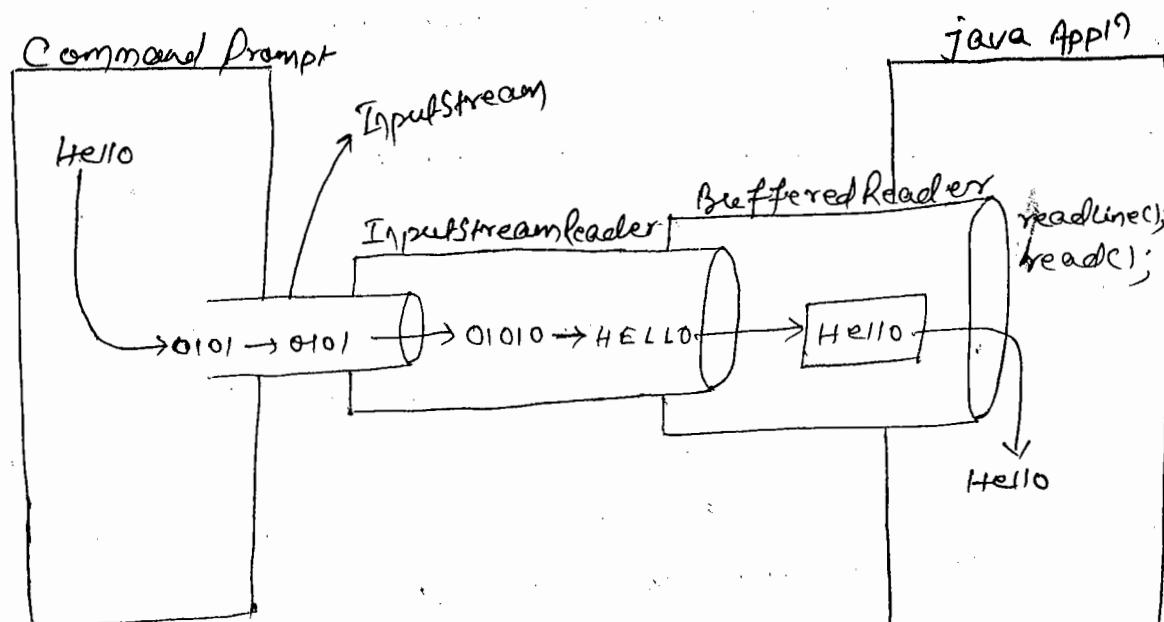
Q. What is the difference between readline() method and read() method?

→ `readLine()` method can be used to read a line of text from `BufferedReader` and return that data in the form of `String`.

public String readLine()

read(1) method can be used to read a single character from BufferedReader and it will return that character in the form of its ASCII value.

public int read()



```
BufferedReader = new BufferedReader(new InputStreamReader(
 (System.in)));
```

Ex:-①

(TSG)

```
import java.io.*;
class BufferedReaderEx1
{
 public static void main(String []args) throws Exception
 {
 BufferedReader br = new BufferedReader (new InputStreamReader
 (System.in));
 System.out.print("Enter Text Data :");
 String data1 = br.readLine();
 System.out.print("Enter the same data again:");
 int val = br.read();
 System.out.println("First Entered Data :" + data1);
 System.out.println("Second Entered Data :" + val + " → " + (char)val);
 }
}
```

Output:-

```
Enter Text Data: Durga Software Solutions
Enter the same data again: Durga Software Solutions
First Entered Data: Durga Software Solutions.
Second Entered Data : 68 →
```



In the case of BufferedReader, if we want to read primitive data or dynamic input then we must use wrapper classes explicitly, because readLine() method is able to read data in the form of String even we enter numeric data on Command prompt.

```

Ex:- import java.io.*;
Class BufferedReaderEx1
{
 public void main (String [] args) throws Exception
 {
 BufferedReader br = new BufferedReader(new InputStreamReader-
 (System.in));
 System.out.print("First Value :");
 String val1 = br.readLine();
 System.out.print("Second Value :");
 String val2 = br.readLine();
 int fval = Integer.parseInt(val1);
 int sval = Integer.parseInt(val2);
 System.out.print("Addition : " + (fval + sval));
 }
}

```

O/P:-

First value: 10

Second value: 20

Addition : 30

Without this  
O/P will be.

====

## ② Scanner:-

In case of BufferedReader to take primitive data of dynamic input then we have to use the following Approach.

- ① Read numeric data in the form of String by using readLine() method.
- ② Convert data from String type to the respective primitive data types by using parseXXX() method from all the wrapper classes.

To Overcome the above problems we have to use Scanner dynamic input approach.

Scanner is a class provided by JDK5.0 version in the form of `java.util.Scanner`, it able to allow to read primitive data directly without using wrapper classes and parseXXX() methods.

→ If we want to use Scanner class in java app's then we have to use the following steps:

### ① Create Scanner Class Object:-

To create Scanner class object we have to use the following constructor from `java.util.Scanner` class.

`public Scanner (InputStream is)`

Ex:- `Scanner s = new Scanner (System.in);`

Z

## (2) Read dynamic input through Scanner class:-

To read primitive data as dynamic input we have to use the following method.

```
public xxx nextxxx()
```

where xxx may be byte, int, short, ...

To read String data as dynamic input we have to use the following method.

```
public String next()
public String nextLine()
```

Ex:-

```
import java.util.*;
class ScannerEx
{
 public static void main(String[] args) throws Exception
 {
```

```
 Scanner s = new Scanner(System.in);
 s.o.p("Employee Number :");
 int cno = s.nextInt();
 s.o.p("Employee Name :");
 string ename = s.nextLine();
 s.o.p("Employee Salary :");
 float esal = s.nextFloat();
 }
```

```

S.O.P ("Employee Address :");
String eaddr=s.next();
S.O.P("Employee Details");
S.O.P("-----");
S.O.P("Employee number :" +eno);
S.O.P("Employee Name :" +ename);
S.O.P("Employee Salary :" +esal);
S.O.P("Employee Address :" +eaddr);
}
}

```

Q. What is the difference between `next()` method and `nextLine()` method?

= `nextLine()` method is able to take a line of text from command prompt in the form of String.

`next()` method is able to read a single word from the provided dynamic input in the form of String.

Ex:-

```

import java.util.*;
class ScannerEx
{
 public static void main(String [] args) throws Exception
 {
 Scanner s=new Scanner (System.in);
 S.O.P("Enter Text Data :");
 String data1=s.nextLine();
 }
}

```

```

Scanner s=new Scanner (System.in);
S.O.P("Enter Text Data :");
String data1=s.nextLine();
S.O.P("Enter the same data again :");

```

```

String data2 = s.next();
System.out.println("First Entered :" + data1);
System.out.println("Second Entered :" + data2);
}
}

```

O/P:-

Enter Text Data : Durga Software Solutions

Enter the Same data Again : Durga Software Solutions.

First Entered : Durga Software Solutions.

Second Entered : Durga (It's come because of s.next() ~~is used~~ is used  
and next() will read only the first word).

### ③ Console:-

In java applications, to take dynamic input if we use BufferedReader and Scanner then we are able to get the following problems.

① For each and every dynamic input we have to use 2 instructions.

① System.out.println(" ") to display request message.

② br.readLine() or s.next(), ... method to read dynamic input.

② X10 security for the data like password data, PIN numbers, ... while entering dynamic input.

To overcome the above problems we have to use Console. Console is a class provided by java version 1.5 in the form of java.io.Console.

→ If we want to use Console in java applications then we have to use the following steps.

## ① Create Console Object:-

To create console class object we have to use the following method from java.lang.System class.

```
public static Console getConsole()
Ex:- Console c = System.console();
```

## ② Get dynamic input through Console:-

To display a request message and to read string data as dynamic input then we have to use the following method.

```
public String readLine(String message)
```

To display a request message and to get read password data as dynamic input then we have to use the following method.

```
public char[] readPassword(String message)
```

Ex:-

```
import java.io.*;
class Console
{
 public void main(String [] args) throws Exception
 {
 Console c = System.console();
 String uname = c.readLine("User Name :");
 char [] pwd = c.readPassword("Password :");
 String upwd = new String(pwd);
 }
}
```

```
if (uname.equals("durga") & upwd.equals("durga"))
{
 System.out.println("Valid User");
}
else
{
 System.out.println("Invalid User");
}
```

### \* Importance of System.out.println() method in java:-

→ The main purpose of System.out.println() method is to display some data on command prompt.

Syntax:-

```
System.out.println(Some data);
```

Where 'Out' is a static reference variable defined in java.lang.System class, it able to refer a predefined java.io.OutputStream, which is connected with command prompt.

→ Where println() is a method defined in OutputStream class, it can be used to send data to OutputStream, where OutputStream will send data to command prompt.

```
public void println(XXX value)
```

Where XXX may be byte, short, int, .....

Q. What is the difference between `println()` method and `print()` method?

→ `Println()` method will display <sup>the specified</sup> data on command prompt and it will keep cursor in the next line.

`Print()` method will display the specified data on command prompt and it will keep cursor in the same time.

Ex:- `Println()` & `Print()`

Class Test

```

{
 public static void main(String[] args)
 {
 System.out.println("Durga");
 System.out.println("Software");
 System.out.println("Solutions");
 System.out.print("Durga");
 System.out.print("Software");
 System.out.print("Solutions");
 }
}

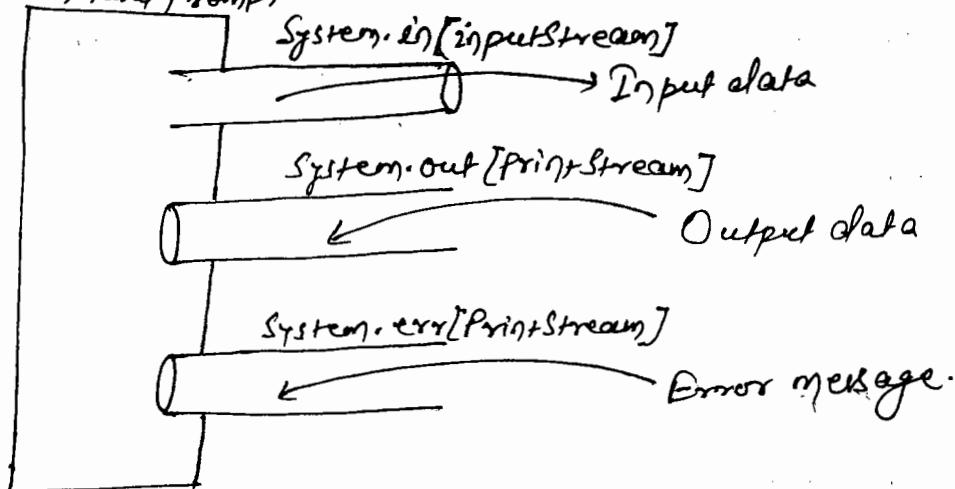
```

`System.out.println(`

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

→ In java applications, to interact with the command prompt for Input and Output Operations, java has provided the following streams.

Command Prompt:



## \* Files in Java:-

File is a memory element to store data.  
There are two types of files in java.

- ① Sequential files
- ② Random Access files.

### ① Sequential files:-

These files are able to allow to read data in sequential manner.

To represent sequential files in java applications, java has provided a separate predefined class that is "java.io.File".

→ To create file class Object we have to use the following constructor.

```
public File(String file-path)
{
 file f2 = new File("abc.txt");
```

- If we use the above instruction to create file then JVM will create only File class object in heap memory, but not file really in folder structure.
- If we want to create file really in folder structure then we have to use the following method.

public File createNewFile()

To get name, parent location and absolute path of the file we have to use the following methods.

public String getName()  
 public String getParent()  
 public String getAbsolutePath()

Ex:-

```
import java.io.*;
class FileEx
{
 public static void main(String[] args) throws Exception
 {
 File f = new File("C:/abc/xyz/cmp.txt");
 f.createNewFile();
 System.out.println(f.isFile());
 System.out.println(f.isDirectory());
 System.out.println("File Name :" + f.getName());
 System.out.println("File Parent :" + f.getParent());
 System.out.println("Absolute Path :" + f.getAbsolutePath());
 FileOutputStream fos = new FileOutputStream(f);
 int size = fos.available();
```

```
byte[] bt = new byte[Size];
fis.read(bt);
S-O.PIN(new String(bt));
fos.close();
fis.close();
}
```

2

If we want to create directory at a particular location we have to use the following method on file class object reference variable.

```
public File mkdir()
```

Ex:- File f = new File("employee");
f.mkdir();

To check whether file is created or not and directory is created or not we have to use the following methods.

```
public boolean isFile()
public boolean isDirectory()
```

To get data of the directory like name, parent location and absolute path we have to use the following methods-

```
public String getName()
public String getParent()
public String getAbsolutePath()
```

2

Ex:-

```

import java.io.*;
Class FileEx
{
 PSVM (String[] args) throws Exception
 {
 File f = new File ("C:/abc/xyz/employee");
 f.mkdir();
 S.O.Pln(f.exists());
 S.O.Pln (f.isDirectory());
 S.O.Pln (f.getName());
 S.O.Pln (f.getParent());
 S.O.Pln (f.getAbsolutePath());
 }
}

```

{}

## (2) Random Access Files:- (Outdated Not Require for present java application)

These files are able to allow to read data from random positions. To represent random access files in java applications, java has provided a predefined class in the form of "java.io.RandomAccessFile".

To create random access file class object, we have to use the following constructor from java.io.RandomAccessFile class.

```
public RandomAccessFile(String file-name, String Access-
privileges)
```

To write data to Random Access file, we have to use the following method.

```
public void writexxx(xxx value)
```

Where xxx may be byte, short, int, UTF[String], ...

→ To move file pointer to a particular position in Random Access file we have to use the following method.

```
public void seek(int position)
```

→ To read data from Random Access file we have to use the following method.

```
public xxx read xxx
```

Where xxx may be byte, short, int, UTF[String], ...

Ex:-

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

```
Class RAFFA
```

```
{
```

```
PSVM (String [] args) throws Exception
```

```
{ RandomAccessFile raf = new RandomAccessFile("emp.txt","rw");
```

```
raf.writeInt(11);
```

```
raf.writeUTF("AAA");
```

```
raf.writeFloat(5000.0f);
```

```
raf.writeUTF("Hyd");
```

```
raf.seek(0);
```

```
S.o.p19 (raf.readInt());
S.o.p19 (raf.readUTF());
S.o.p19 (raf.readFloat());
S.o.p19 (raf.readUTF());
```

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## NETWORKING

## Networking:-

By using java programming language, we are able to prepare the following two types of applications.

- ① Standalone Applications
  - ② Distributed applications.

## ① Standalone Applications:-

These applications are designed without using client-server Arch. To prepare these applications we have to use java core library like `java.io`, `java.util`, `java.lang`, ... packages.

## ② Distributed Applications:-

Distributed Applications:-  
If we design any java application on the basis of client-server arch then that java appli is called as distributed application.

To prepare Distributed Applications, Java has provided the following technologies.

① Socket Programming

② RMI

③ CORBA

④ EJBs

⑤ WEBSERVICES

① Socket Programming :-

→ To prepare distributed applications if we use socket programming then we have to establish sockets between local machine and remote machine ~~between~~ on the basis of 4th IP-Address and port numbers.

Q. What is the difference between IP address and port numbers?

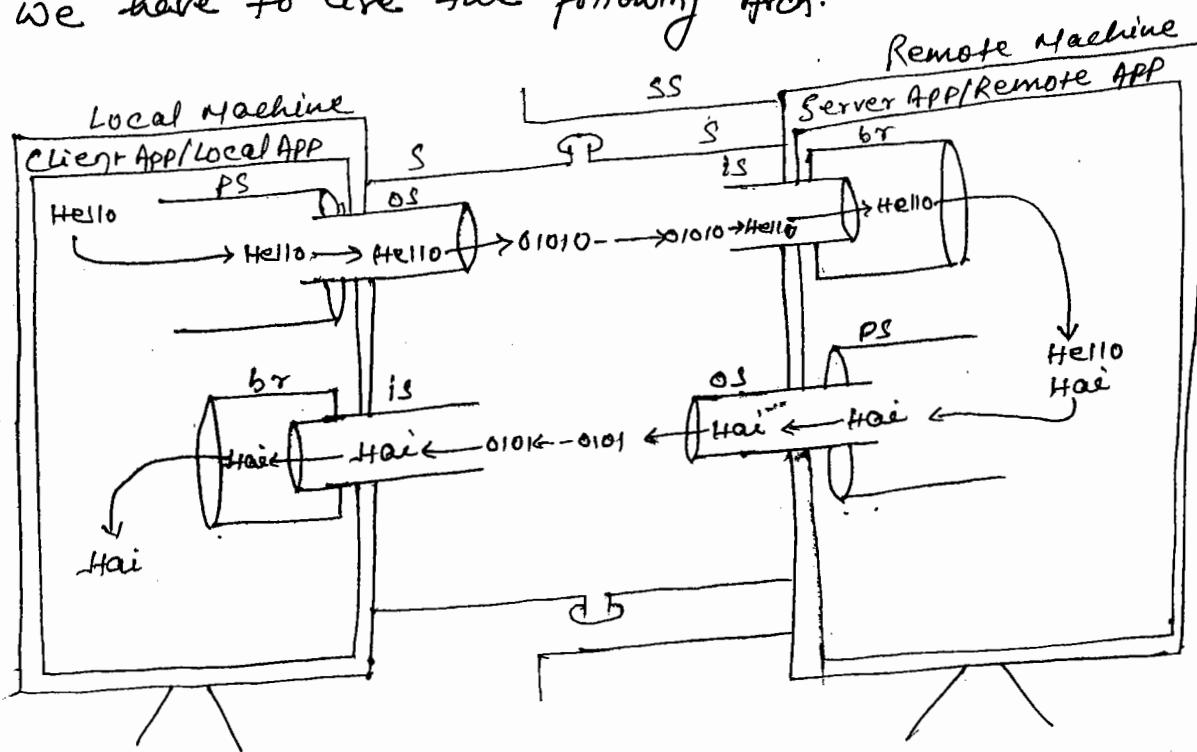
→ IP - Address is an unique identity provided to each and every machine over the network, it would be provided by the network manager at the time of network configuration.

Port number is an unique identity provided to each and every process being executed within a single machine, it would be provided by local Operating System.

→ Socket is a medium or channel, it able to transfer the data from one machine to another machine over the network.

To prepare distributed ~~socket~~ app? by using socket progng, JAVA has provided predefined library in the form of "java.net" package.

→ To prepare Distributed App by using socket programming we have to use the following Arch.



### Steps to design Client Application:-

- ① Create Socket class object in order to send connection establishment request to Server Socket :-

To create Socket class Object we have to use the following Constructor.

```
public Socket(String server_IP_Addr, int Server_Port_Number)
```

Ex:- `Socket S = new Socket('localhost', 4444);`

- ② Get OutputStream from Socket :-

To get OutputStream from Socket we have to use the following method from `java.net.Socket`.



public OutputStream getOutputStream()

Eg:- OutputStream OS = s.getOutputStream();

③ Create PrintStream with OutputStream :-

PrintStream PS = new PrintStream(OS);

④ Send data to PrintStream :-

```
String data = "Hello";
PS.println(data);
```

With the above steps, data will be transferred from local machine to remote machine, where remote will process request and sends some response to local machine, now, response data is available at client's side socket.

⑤ Get InputStream from Socket :-

To get InputStream from Socket we have to use the following method from java.net.Socket class.

public InputStream getInputStream()

Eg:- InputStream is = s.getInputStream();

⑥ Convert InputStream into BufferedReader :-

BufferedReader br = new BufferedReader(new InputStreamReader(is));

⑦ Read data from BufferedReader:-

```
String data = br.readLine();
System.out.println(data);
```

\* Steps to create Server Side Application:-

① Create Server Side Socket:-

To Create Server Side Socket we have to create Object for ServerSocket class with the following constructor.

```
public ServerSocket (int port-number)
```

Ex:- ServerSocket ss = new ServerSocket(4444);

② Accept the connection establishment request from Client Socket:-

To accept the client side socket connection establishment request we have to use the following method from java.net.ServerSocket class.

```
public Socket accept()
```

Ex:-

```
Socket s = ss.accept();
```

③ Get InputStream from Socket:-

```
InputStream eis = s.getInputStream();
```

④ Convert InputStream eis to BufferedReader:-

```
BufferedReader br = new BufferedReader(new InputStreamReader(eis));
```

⑤ Read request data from BufferedReader:-

```
String data = br.readLine();
System.out.println(data);
```

## ⑥ Get OutputStream from Socket:-

OutputStream os = s.getOutputStream();

## ⑦ Convert OutputStream into printStream:-

PrintStream ps = new PrintStream(os);

## ⑧ Send Response data to printstream:-

String data = "hai";  
ps.println(data);

## Ex:- Client App

```
import java.io.*;
import java.net.*;
public class ClientApp
{
 public static void main(String[] args) throws Exception
 {
 Socket s = new Socket("localhost", 4444);
 OutputStream os = s.getOutputStream();
 PrintStream ps = new PrintStream(os);
 BufferedReader bri = new BufferedReader(new InputStreamReader(
 System.in));
 String data1 = bri.readLine();
 ps.println(data1);
 InputStream is = s.getInputStream();
 BufferedReader bre = new BufferedReader(new InputStreamReader(
 is));
 String data2 = bre.readLine();
 System.out.println(data2);
 }
}
```

## ServerApp.java

```

import java.io.*;
import java.net.*;
public class ServerApp
{
 public void (String []args) throws Exception
 {
 ServerSocket ss = new ServerSocket(4444);
 Socket s = ss.accept();
 InputStream is = s.getInputStream();
 BufferedReader br1 = new BufferedReader(new InputStreamReader(is));
 String data1 = br1.readLine();
 System.out.println(data1);

 OutputStream os = s.getOutputStream();
 PrintStream ps = new PrintStream(os);
 BufferedReader br2 = new BufferedReader(new InputStreamReader(System.in));
 String data2 = br2.readLine();
 ps.println(data2);
 }
}

```

Above application is providing single communication.

Following Application are providing infinite communication.

Ex:- ClientApp.java (for infinite communication).

```
import java.io.*;
import java.net.*;
public class ClientApp
{
 public static void main(String[] args) throws Exception
 {
 Socket s = new Socket("localhost", 4444);
 OutputStream os = s.getOutputStream();
 PrintStream ps = new PrintStream(os);
 BufferedReader br1 = new BufferedReader(new InputStreamReader(
 System.in));
 InputStream is = s.getInputStream();
 BufferedReader br2 = new BufferedReader(new InputStreamReader(
 is));
 while (true)
 {
 String data1 = br1.readLine();
 ps.println(data1);
 String data2 = br2.readLine();
 s.getOutputStream().println(data2);
 if (data1.equals("bye") && data2.equals("bye"))
 {
 System.exit(0);
 }
 }
 }
}
```

## ServerApp.java (for infinite communication)

(149)

```
import java.io.*;
import java.net.*;
public class ServerApp
{
 public void main(String [] args) throws Exception
 {
 ServerSocket ss = new ServerSocket(4444);
 Socket s = ss.accept();
 InputStream is = s.getInputStream();
 BufferedReader br1 = new BufferedReader(new InputStreamReader(
 Reader(s))); - Reader(is));
 OutputStream os = s.getOutputStream();
 PrintStream ps = new PrintStream(os);
 BufferedReader br2 = new BufferedReader(new InputStreamReader(
 Reader(System.out))); - Reader(System.out));
 while(true)
 {
 String data1 = br1.readLine();
 s.getOutputStream(data1);
 String data2 = br2.readLine();
 ps.println(data2);
 if(data1.equals("bye") && data2.equals("bye"))
 {
 System.exit(0);
 }
 }
 }
}
```

20/09/2015

## RMI (Remote Method Invocation) :-

To design distributed applications if we use socket programming then we are able to get the following drawbacks.

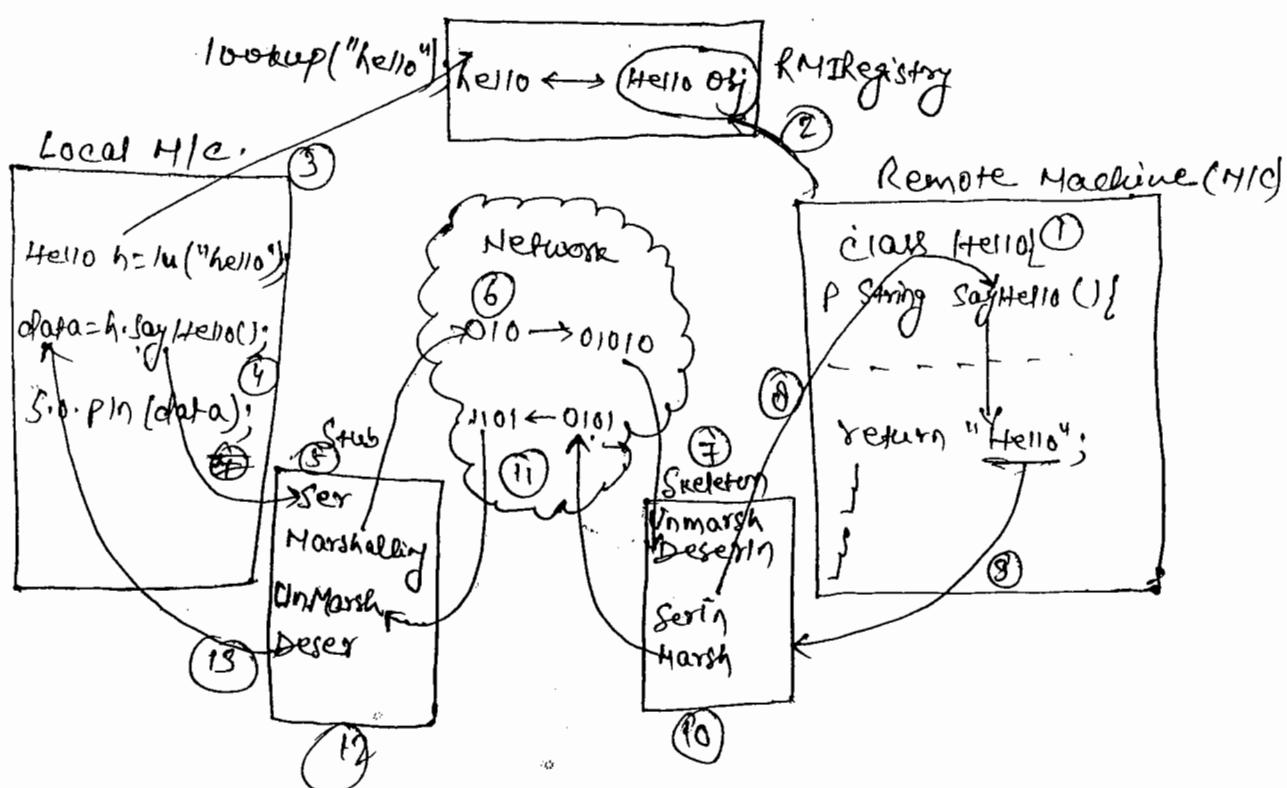
- ① Developers must create client side socket and server socket at server machine explicitly.
- ② Developers must create InputStreams and OutputStreams with the socket explicitly.
- ③ If we want to transfer an object from one machine to another machine then we have to perform serialization and deserialization explicitly.
- ④ To overcome all the above problems in distributed applications we have to go for RMI.

above specified applications if we use RMI then the components like "Stub" and "Skeleton".  
→ Where "Stub" is a special software available at local machine and it able to provide the required sockets, Inputstreams and OutputStreams, Marshalling and Unmarshalling at Local Machine.  
Where "Skeleton" is a special software available at Remote machine, it will provide Server socket, socket, Inputstreams and OutputStreams, Serialization and Deserialization, Marshalling and Unmarshalling and it able to access remote methods.

→ In RMI applications, we have to define Remote method at Remote machine and we have to access that remote method from Client machine. To access Remote method from Client machine, first we have to make available Remote Object to client machine, for this we have use a Special software called as "RMIREGISTRY".

RMIREGISTRY is an external tool provided by JAVA, it able to manage all the ~~data~~ Remote objects along with their logical names in the form of name-value pairs, that is, RMIREGISTRY will expose all the remote objects in the network.

To prepare RMI Applications, we have to use the following architecture.



- ① Prepare remote class with remote method at Remote machine which we want to access from local machine.
- ② Keep Remote Object in RMI Registry in order to expose Remote Object in Network.
- ③ Perform "Lookup" Operation over RMIREGISTRY on the basis of the logical name, where RMIREGISTRY will search for the required remote object and it will send the remote object reference to Client Application as the response to Lookup Operation.
- ④ Access Remote method from Client Application.
- ⑤ Stub will take Remote method call, Stub will perform the required serialization over parameters and Marshalling.
- ⑥ Stub will send Remote method call to network, where network will carry Remote method call to Remote machine.
- ⑦ At Remote machine, Skeleton will take Remote method call, Skeleton will perform the required Unmarshalling and Deserialization over parameters.
- ⑧ Skeleton will access Remote method.
- ⑨ After executing the Remote method, Skeleton will take the return value from remote method.
- ⑩ Skeleton will perform the required Serialization and marshalling over return value.
- ⑪ Skeleton will send the return value to network, where network will carry return value to Local machine.
- ⑫ At Local machine, Stub will take the return value from network and Stub will perform the required

Unmarshalling and Deserialization over the return value.

(3) Stub will send the return value to Client Application.

### Steps to Prepare RMI Application:-

- (1) Create Remote Interface.
- (2) Create Remote Interface implementation class.
- (3) Create Registry Software or Program.
- (4) Create Client Application.

#### ① Create Remote Interface:-

The main interface of Remote interface is to declare all the remote methods.

Steps:-

- (a) Declare an User defined Interface.
- (b) Extend User defined Interface from java.rmi.Remote interface.
- (c) Declare all remote methods with throws java.rmi.RemoteException.

Ex:- D:\javab\rmiclass\HelloRemote.java

Ex:-

```
public interface HelloRemote extends java.rmi.Remote
```

```
 public String sayHello(String name) throws java.rmi.RemoteException;
```

#### ② Create Remote Interface Implementation Class:-

The main purpose of creating implementation class for remote interface is to provide implementation for all the service methods declared in the remote interface.

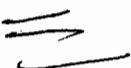
### Steps:-

- (a) Declare an User Defined Class.
- (b) Extends User defined class from `java.rmi.server.UnicastRemoteObject` abstract class inorder to make eligible to keep Remote Object in RMI Registry.
- (c) Implement remote interface in user defined class.
- (d) Declare a public and 0-arg constructor with throws `java.rmi.RemoteException`.
- (e) Implement all the Remote interface methods in the implementation class.

Ex:- D:\javag6\rmil\app1\HelloRemoteImpl.java

Ex:-

```
public class HelloRemoteImpl extends java.rmi.server.UnicastRemoteObject implements HelloRemote {
 public HelloRemoteImpl() throws java.rmi.RemoteException {
 //
 }
 public String sayHello(String name) throws java.rmi.RemoteException {
 return "Hello.... "+name+"!";
 }
}
```



### ③ Create Registry Software:-

The main purpose of registry program is to create remote Object and to keep Remote Object in RMIRegistry with a particular logical name.

Steps:-

- Create an User defined class with main() method.
- Inside main() method, Create Object for remote interface implementation class.
- Bind remote Object with a logical name in RMIRegistry by using the following method.

public static void bind(String logical\_name, Remote obj).  
throws RemoteException.

Ex:- D:\javab6\rmil\app1\HelloRegistry.java

Ex:-

```
public class HelloRegistry
{
 public static void main (String [] args) throws Exception
 {
 HelloRemote hr = new HelloRemoteImpl ();
 java.rmi.Naming.bind ("Hello", hr);
 System.out.println ("HelloRemote Object is binded with 'Hello' "
 + "logical name in RMIRegistry");
 }
}
```

Z

## ④ Create Client Application:-

The main purpose of Client application is to get Remote Object from RMIREGISTRY and to access remote methods.

Steps:-

① Declare an user defined Class with main() method.

② Perform "lookup" Operation in RMIREGISTRY for Remote Object by using the following method from java.rmi.Naming class.

public static Remote lookup(String logical-name) throws  
RemoteException.

③ Access Remote methods.

Ex:- D:\Java\rmi\app\|ClientApp.java

Ex:-

```
public class ClientApp
{
 public static void main(String[] args) throws Exception
 {
 HelloRemote hr = (HelloRemote) java.rmi.Naming.lookup("Hello");
 String data = hr.sayHello("Durga");
 System.out.println(data);
 }
}
```

## Execution process:-

D:\javab\rmilapp1> set classpath=.;  
 D:\javab\rmilapp1> javac \*.java  
 D:\javab\rmilapp1> start RMIREGISTRY  
 D:\javab\rmilapp1> start java HelloRegistry  
 D:\javab\rmilapp1> java ClientAPP

## D:\javab\rmilapp2

- CalculatorRemote.java
- CalculatorRemoteImpl.java
- CalculatorRegistry.java
- ClientAPP.java

### Ex:- CalculatorRemote.java

```
import java.rmi.*;
public interface CalculatorRemote extends Remote {
 public int add(int i, int j) throws RemoteException;
 public int sub (int i, int j) throws RemoteException;
 public int mul (int i, int j) throws RemoteException;
}
```

### Ex:- CalculatorRemoteImpl.java

```
import java.rmi.*;
import java.rmi.server.*;
public class CalculatorRemoteImpl extends UnicastRemoteObject
- Object implements CalculatorRemote,
```

```

 {
public CalculatorRemoteImpl() throws RemoteException
{
}

public int add(int i, int j) throws RemoteException
{
 return i+j;
}

public int sub(int i, int j) throws RemoteException
{
 return i-j;
}

public int mul(int i, int j) throws RemoteException
{
 return i*j;
}
}

```

### For:- CalculatorRegistry.java

```

import java.rmi.*;
public class CalculatorRegistry
{
 public void main(String[] args) throws Exception
 {
 CalculatorRemote cr = new CalculatorRemoteImpl();
 Naming.bind("Cal", cr);
 System.out.println("CalculatorRemoteImpl Object is binded with
 'Cal' logical name in RMIREgistry");
 }
}

```

Ex:- ClientApp.java

```

import java.rmi.*;
public class ClientApp
{
 public void main(String[] args) throws Exception
 {
 CalculatorRemote cr = (CalculatorRemote) Naming.lookup("cal");
 System.out.println("ADD :" + cr.add(10, 5));
 System.out.println("SUB :" + cr.sub(10, 5));
 System.out.println("MUL :" + cr.mul(10, 5));
 }
}

```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

2  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.