

## Java Technology

- 1) It is a programming language.
- 2) It is a platform.

Java Programming language is a basic building block of Java Technology.

**Fundamentals Of Java---OCJA(Oracle Certified Java Associate)**

**Core Java-----OCJP(Oracle Certified Java Programmer or Professional)**

**Advance Java-----OCEWCD(Oracle Certified Expert Web Component Developer)**

**Java Programming Language = Fundamentals Of Java(OCJA) + Core Java(OCJP)  
(Java Interview=80% Questions)**

## Advance Java

Servlet  
JSP---Java Server pages  
Java Beans  
Enterprise Java Beans  
Framework-----Struts, Hibernate, Spring

## History of Java

Sun Micro System----Research Project----To develop new programming language for consumer electronic devices like TV, VCR, Tap Recorder, Washing Machine.

Green Team----Team Leader----James Gosling-----Window---OAK Tree

**First name of java is OAK language.**

Market Research-----Internet----Static Website----Dynamic Website

## Java Features

**1) It is a platform Independent Language.**

Platform---Hardware + OS (Operating System)

## Platform Dependent

**Developed By Jaysukh Patel**

C Program(first.c)-----Windows+Hardware--Compile---Machine Code(Object Code or 0 and 1)--first.obj--C Linker---C Library-----first.exe(Executable File)---Platform Dependent

first.exe---Linux + Hardware-----Does not Execute.

first.exe---Sun Solaris + Hardware-----Does not Execute.

first.exe---Mac + Hardware-----Does not Execute.

### Platform Independent

Java Program(first.java)---Windows + Hardware—Compile---ByteCode (Not a 0 and 1-Intermediate Code-Platform Independent Code)---Code Of JVM---first.class

first.class(ByteCode)—Linux + Hardware(Platform Dependent JVM)-(Machine Code)-execute

first.class(ByteCode)—Sun Solaris + Hardware(Platform Dependent JVM)-(Machine Code)-execute

first.class(ByteCode)—Mac + Hardware(Platform Dependent JVM)-(Machine Code)-execute

**JVM---Java Virtual Machine--Platform Dependent**

**JRE--Java Runtime Environment--Platform Dependent**

**JVM**

**Java API(Application Programming Interface) or Java Class Library(JCL)**

**JDK--Java Development Kit---Platform Dependent**

Java compiler

Java Interpreter(JRE--JVM)

JVM

Java API(Application Programming Interface) or Java Class Library(JCL)

Java Debugger

Jar File Tools (Java Archive File---JAR)

**2)It is a Pure Object Oriented Language.**

C---Procedure Oriented Language or Structured Programming Language or Modular Programming Language.

**Disadvantages Of C Language**

Functions Communicate with each other---Data is freely flow able and accessible.  
There is no data security.

Main Focus----Sequence of Steps

Large Programs----

- Error finding becomes difficult

- Debugging becomes difficult

- Future Modification becomes difficult

- Maintenance becomes difficult

**Object Oriented Programming-----**That is based on real life objects.

Main Focus-----Object-----Object contains data and functions.

It provides data security.

Objects communicate with each other—via function calling

Whole program is divided into the no of objects—and each object contains data and functions

Large Programs----

- Error finding becomes easy

- Debugging becomes easy

- Future Modification becomes easy

- Maintenance becomes easy

**Simula67---**Object Oriented Programming Language.

**C++----**It is an extension of C Language.

**C++----**It is half pure Object Oriented Programming Language.

**Java---** It is a pure Object Oriented Language.

**OOPS Concepts(Object Oriented Programming System Concepts)**

**1)Data Encapsulation**

Wrapping of data and methods into a single unit is called Data Encapsulation.

**2)Data Hiding**

**Two reasons to hide the data**

- 1)To provide the security

- 2)To reduce the complexity of the object

## 3)Inheritance

new object = exiting + new features

object----data and functions---existing object

new object-----new data and functions

**Inheritance provides the re usability of code.**

### Advantages

- 1)Duplicate coding--- reduce
- 2)Program length--- reduce
- 3)Memory saving
- 4)Time saving

## 4)Polymorphism

It is an ability to take more than one form.

One thing multiple forms or one thing multiple behavior.

Method overloading or Function overloading-----Example of Polymorphism

## 5)Data Abstraction

It is an act of representing essential features without including background details.

### 3)It provides the security(It is a secured language)

#### 1)Java Is a pure Object Oriented Programming Lang.

Data Hiding-----Data Security

Data Abstraction-----Data Security

#### 2)JVM---Java Virtual Machine

1)ByteCode---Memory Load

2)ByteCode Verification-----Check It is legal or not(For Security)

3)Machine Code----Execute

#### 3)In C and C++-----int is compatible with boolean.

any positive or negative no-----true

zero-----false

if(10)---true

if(-10)---true

if(0)---false

**In Java---int is not compatible with boolean.**

Java introduce new data type---boolean-----true or false

if(10)---compilation error

if(true)---OK

**4)In C and C++-----Pointer-----Direct Memory Access----Data is not Secured**

Remove the concept of pointer and introduce the new concept of reference.

**4)It provides the concept of Mutithreading.**

## **Multiprocessing**

cpu1,cpu2,cpu3-----more than one CPU

p1,p2,p3-----more than one processes

at the same time-----execute

## **Multitasking(Heavy Weight Processing)**

cpu1-----only one CPU

p1,p2,p3-----more than one processes

at the same time-----execute

CPU-----time sharing and fast switching

## **MultiThreading(Light Weight Process)**

cpu1-----only one CPU

p1-----one process or java program(Multiple task)

at the same time-----execute

CPU-----time sharing and fast switching

**Java Program-----t1----1 to 10**

t2----11 to 20

t3----21 to 30

single process and single thread

1-----10 11-----20 21-----30

## **MultiThreading----single process and multiple thread**

t1-----1 to 10

t2-----11 to 20

t3-----21 to 30

1 11 21 2 12 22 3 13 23-----

**applet**-----client side execution

**Servlet**-----server side applet-----server side execution-----MultiThreading

web based application----CGI programs----(Common Gateway Interface)  
for each request-----separate process

Servlet-----for each request-----separate thread---only one process

## **5) Java Programs are distributed on network easily**

jar file-----Java archive file-----

executable jar file-----

library jar file-----you can use these jar file as a library

## **Product Life Cycle(PLC)**

### **Software Development Life Cycle(SDLC)**

#### **7 Stages**

1)analysis and requirement gathering

2)design and blue print creation

3)Actual Development and coding

4)Testing-----Manual Testing and automation testing

5)Implementation

6)Maintenance

7)End of life

## **Object Oriented Programming**

### **Object Oriented analysis**

## 1)To identify the two types of objects in your system.

- 1)physical object
- 2)conceptual object

**physical object**----Student,Employee,Book

**conceptual object**---order, transaction

## 2)To identify the properties of each object.

Properties or attributes or data or variables

**Student**-----rollno,name,address,Phoneno

**Employee**---code no,department,salary

**Order**----order id, order date, product description, price, quantity

**Transaction**---trasacid, trndate,trnamt

## 3)To identify the functions of an objects.

functions or methods or operations or behavior

**Student**-----getData(),showData(),getPer()

## OOPS Concepts

### 1)Data Encapsulation & Data hiding

**Data Encapsulation**-----classes and objects

**Data Hiding**-----Access Specifier or Visibility Mode

Private-----It can be accessible in the same class

no specifier(default—package private)---It can be accessible in the same package

protected-----same package(yes)+outside the package- only in child

public-----at any place

**class**:---It is a general form or blueprint or template which defines the shape and behavior of an object.

**shape**-----properties or attributes or data or variables

**behavior**---functions or methods or operations

**object**—Once a class has been defined you can create a physical instance of that class and that is called an object of that class.

## Class declaration

```
Access Specifier class classname
{
    properties or attributes or data or variables
    functions or methods or operations or behavior
}
```

**outer class**-----public and default

**inner class**----private,default,protected,public

## How to create Object of any class

### 1) by using new

new classname();-----unreferenced object

```
new classname().showdata();
new classname().getdata();
```

### 2)

```
classname variable;
(Object reference variable----It can store the address of object----remote)
variable=new classname();
```

```
variable.getdata();
variable.showdata();
```

### 3)classname variable=new classname();

```
variable.getdata();
variable.showdata();
```

```
new Student();
```



```
Student s1;  
s1=new Student();
```

```
Student s1=new Student();
```

**you can create multiple reference of the same object.**

```
Student s1=new Student();  
Student s2=s1;  
Student s3=s2;  
Student s4=s1;
```

```
s1.rollno=10-----s2.rollno,s3.rollno,s4.rollno-----10
```

**c++-----Memory Management -----programmer-----coding**

**c-----**malloc, calloc, realloc, free  
dynamic memory allocation and deallocation

**c++-----**new and delete-----dynamic memory allocation & deallocation

**Garbage collection-----**It is an automatic memory management technique.

unreferenced object or unused object-----identify-----memory free

1)

```
Student s1=new Student();  
Student s2=new Student();  
Student s3=new Student();  
s3=s1;  
s1=s2;  
s2=null  
s3=s1;
```

How many objects are eligible for garbage collection

2)

```
Student s1=new Student();
Student s2=new Student();
Student s3=new Student();
s3=s1;
s1=new Student();
s2=s1
s3=new Student();
s2=s3;
```

How many objects are eligible for garbage collection

3)

```
Student s1=new Student();
Student s2=new Student();
Student s3=new Student();
s3=s1;
s1=s2
s2=s3
s1=new Student();
s2=new Student();
s1=s2
```

How many objects are eligible for garbage collection

## Data Types in Java

**4 categories**

**8 types**

### 1---Numeric

**byte**----- -128 to +127

**short**----- -32768 to +32767

**int**-----

**long**-----

byte and short-----it is not used in computer application.

### 2----Textual

char-----it can store only one character

## 3-----real value

float

double-----higher accuracy

## 4-----boolean

boolean-----true or false

**String---** It is not a data type in java.  
It is a predefined class from java.lang package-----default package  
so you can access all the classes from lang package directly.

## To compile Java Program

```
javac filename.java
```

## To run Java Program

```
java classname-----which contains main function.
```

## How to take Input From user

### 1)Formatted Input

```
import java.util.*;  
Scanner s=new Scanner(System.in);
```

```
s.nextInt()  
s.nextFloat()  
s.next()  
s.nextLong()  
s.nextDouble()
```

### 2)

```
import java.io.*;  
DataInputStream d=new DataInputStream(System.in);  
  
String s=d.readLine();
```

## Data types in Java

byte  
short  
int  
long  
float  
double  
char  
boolean

## Wrapper class

Byte  
Short  
Integer  
Long  
Float  
Double  
Character  
Boolean

```
int a=Integer.parseInt(no in string format)
```

```
int a=Integer.parseInt(s)
float a=Float.parseFloat(s)
long a=Long.parseLong(s)
double a=Double.parseDouble(s)
```

readLine() method of DataInputStream class throws IOException and It is a checked exception so you must catch it or declared to be thrown. Otherwise java compiler will give compilation error.

3)

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

```
String s=d.readLine();
```

```
int a=Integer.parseInt(no in string format)
```

```
int a=Integer.parseInt(s)
float a=Float.parseFloat(s)
long a=Long.parseLong(s)
double a=Double.parseDouble(s)
```

readLine() method of BufferedReader class throws IOException and It is a checked exception so you must catch it or declared to be thrown. Otherwise java compiler will give compilation error.

## Explain System.out.println()

**System**-----It is a predefined class in java.lang package. And lang package is default package in java so no need to import.

**out**-----It is an object reference of PrintStream class from java.io package.

**out**---It is a static member of System class. So you can access out member by just using classname.

## System.out

```
class System
{
    static PrintStream out=new PrintStream();
}
```

**println()**----It is a method of PrintStream class. So you can call this method by using out.

## Explain

**public static void main(String arg[])**-----you can give any name

**or**

**static public void main(String []arg)**

It is a starting point of program execution

**public**-----Access Specifier-----It can be accessible from any place  
It must be inside the class.

private  
protected  
default

compilation error-----no error

run time error-----yes-----main method is not found in class

**static----** you must make main function static  
you can call any static function of any class by just using a classname.  
You need not have to create the object of that class.  
If it is not static then jvm must create the object of that class.

Student.main()

## To compile Java Program(javac----Java Compiler)

javac filename.java

first.java-----compile-----first.class(bytecode)

## To run or execute java program(java---Java Interpreter--JVM)

java classname----which contains main function.

Javac Student.java

java Student

### if it is not static

compilation error----- no error

run time error----- main method is not static in class

A---default

B---default

C---default-----main

A.java or B.java or C.java

javac A.java or javac B.java or javac C.java

java C

A---public

B---default

C---default-----main

A.java

javac A.java

java C

A---public

B---public

C---default-----main

not possible in a single file.

You can create two file

A—public-----A.java

B-- public-----B.java

C---default-----main

javac A.java and javac B.java

java C

**void---** return type----each and every function in java must have return type.  
It does not return any value so it's return type is void

**any other return type**  
int or String-----

compilation error----- no error

run time error----- yes---main method must return a value of type void

**main----**It is fun name

**main(String arg[])**-----command line arguments or run time arguments

**main function of c and c++-----**void main(int argc,char \*argv[])

argc-----argument count  
argv-----argument vector

## any other argument in main function

main(int I)  
main(String s)

compilation error-----no error  
run time error-----yes---main method is not found in class

## can you overload the main function?

**Yes you can overload the main function**

**fun name----same  
but parameter must be different.**

Method overloading can be possible in single class or in inheritance  
return type may be of any type access specifier may be any type----It does not matter

- 1)no of parameter or
- 2)type of parameter
- 3)sequence of parameter

## Opearators in Java

### 1)Arithmetic Opearator

+, -, \*, /, %

int a=10,b=20;

int c;

c = a + b;---30



$c = a - b$  --- -10

$c = a * b$  --- 200

$c = a / b$  -- Quotient

$c = a \% b$  ---Reminder

**Division---Quotient---int/int---result is int**

27/5-----5

int / float----result is float

float / int---result is float

27 / 5.0f----5.4-----

27.0f / 5----5.4-----

**Modulus---Reminder**

int % int---result is int

27 % 5-----2

int / float----result is float

float / int---result is float

27 % 5.0----2.0

27.0 % 5----2.0

First Operand(sign)---Result Sign

27%5----2

-27 % 5-----(-2 )

27 % -5----- ( 2 )

## 2)Comparasion Opearator or Relational Opearator

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

int a = 10,b = 20

a == b----false

a = b(Assignment Operator)

## 3) Logical Opearator(Between Different Conditions)

And --- && (Short Circuit And) or & (Logical and Operator)

Or-----|| (Short Circuit Or) or | (Logical Or)

Not-----!

Ex or-----^

Condi1	Cond2	And	Or	Ex-or
False—0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

And----All the conditions must be true----Result is True

Otherwise ---false

Or—Any one condition must be true----True

Ex-or----The condition must be exclusive(Different) then the result is true

Not-----0---1

1---0

```
int a = 10,b = 20;
```

```
!(a == b)
```

### Short Circuit And ( && )

```
c1 && c2 && c3 && c4 && c5
```

### Short Circuit Or ( || )

```
c1 || c2 || c3 || c4 || c5
```

### Bitwise Operator ( 0 and 1)

1 Byte==8 Bits

00101010---1 byte

Bitwise and----&

Bitwise or-----|

Bitwise Complement-----~

Bitwise left shift----<<

Bitwise right shift- >>

```
int a = 5,b = 8;
```

```
int c = a & b;
```

c----result---0

### Increment Opearator and Decrement Opearator

++

--

```
int a = 10;
```

a = a + 1;-----11---a++(Post Increment) or ++a(Pre Increment)

a = a - 1-----9----- a--(Post Decrement) or --a(Pre Decrement)

int a = 10, b = 20;

int c;

1)

c = a++ + b++;

a = 11      b = 21    c = 30

2)

c = ++a + ++b;

a = 11    b = 21    c = 32

int b = 20;

int a = b++;-----a=20

b=21

int a = ++b;

a = 22

b = 22

## Decision Making Structure or Control Structure

### 1)Simple IF

**Condition:- It must be Boolean value or expression**

```
if(condition)
```

```
{
```

```
    //statments
```

```
}
```

```
--next statments
```

If-----Check Condition-----True---Execute all statements----Next statement  
false-----Next statements

### 2)IF - ELSE

```
if(condition)
```

```
{
```

```
    //statments
```

```
}
```

```
else
```

```
{
```

```
    //statments
```

```
}
```

```
Next statment
```

If---Check Condition---True----Execute if statements---Next

False---Execute else statements--Next

### 3)Ladder IF-ELSE---To check series of condition or multiple conditions

```
if(condition)
{

}

else if(condition)

{

}

else if(condition)

{

}

else

{

}

-Next
```

### 4)Nested IF ELSE

One if else within another if else.

```
if(condition)

{

    if(condition)
```

```
{  
    }  
else  
    {  
    }  
}  
else  
{  
    if(condition)  
    {  
    }  
    else  
    {  
    }  
}
```

Next

### **Ternary Operator or Conditional Operator**

Condition ? Exp1 : Exp2;

True-----Exp1;

False-----Exp2;

## Switch

To test for multiple conditions.

Alternative of ladder if else.

Equality check only.

You can pass only int or character or enum expression in switch(up to Java 1.6)

You can pass String in switch(Java 1.7)

switch(int or character or enum or String)

```
{  
    case value:statements;  
        break;  
    case value:statements;  
        break;  
    case value:statements;  
        break;  
    default:statements;  
        break;  
}
```

## Loop Structure or Iterative Structure

To do any task repeatedly.

**1)do while**



## 2)while

## 3)for

## 1)do while

It is post tested loop or exit controlled loop.

It must be executed at least once when condition is false.

do

{

Statements

-----

-----

}

While(boolean condition);

Next

**Do-----Execute all statements----while—Check Condition----True-----execute all statements---while—check condition---False---next**

## 2)while loop

It is pre tested loop or entry controlled loop.

while(boolean condition)

{

Statements

```

-----
-----
}

```

Next

**while---check condition-----true---execute all statements---while---Check  
Codition---False---next**

### 3)for loop

It is pretested loop or entry controlled loop.

For(Initialization; Boolean Condition; Expression)

```

{
    Statements
    -----
    -----
}

```

**for--Initialization---Check Condition--True--Execute Statments--Expression-  
Check Condition--True--Execute Statments--Expression--Check Condition--False--next**

for( ; ; )-----Infinite Loop

```

{
    //statements
}

```

```
for( ;condition;)

{

    //statment

}

for(i=1,j=10,k=5; i<=10; i++,j--,k++)

{

}


```

**fixed value-----for**  
**otherwise—do--while or while**

**Any method contains for parts**

```
Return type    Method Name(Argument List or Parameter List)
{
    //body of method
}


```

1)Without Return Type Without Parameter

```
void getdata();
```

2)Without Return Type With Parameter

```
void getinfo(int r,Stinrg n,String a,long p)
```

3)With Return Type Without Parameter

```
int getrollno()
{
    Return rollno;
}


```

4)With Returntype With Parameter

```
int sum(int ,int );
```

## Java API or Java Packages

java.lang----default package  
                  System, Wrapper class, Math class  
java.io-----Input/Output stream---IO related operation  
java.util----- Utility class-----Collection, Scanner, Date  
java.net----- Networking related classes  
java.sql----- Database connectivity(JDBC) related classes  
java.awt---- Abstract Window Toolkit----GUI based application related classes  
javax.swing Advance component in GUI

```
import java.util.*;  
Or  
import java.util.Scanner;  
import java.util.subapckage.*;
```

## Types of variables in java

### 1)Local Variable

The variables declared inside any method is called local variable of that method.

**Scope**—In the method in which it is declared.

**Lifetime**---at the end of method execution.

### 2)Instance Variable

The variables declared inside the class but outside the method is called instance variable of class.

It is a property of an instance or object.

Each object have an individual copy of instance variable.

**Scope**---in the same class in which it is declared.

From outside the class it depends on access specifier.

You can access any instance variable directly inside the class.

Form outside the class you must required any object.

**Life time**-----at the end of object destruction.

Instance variable can not be accessible in static method without object reference.

## 3)Static Variable or Class Variable

The variables declared inside the class with static keyword is called static variable or class variable

It is a property of class not an object.

Only one copy of static variable is created for entire class.

So each and every object can share this variable and can change or access the value of the variable.

It can be accessible directly in the same class.

From outside the class you must required classname to access the static variable.

Static variable can be accessible in instance method.

## Default Values

All the instance variables and static variables are automatically initialized by their default values

int----0

float—0.0

String-----null

Any reference----null

boolean----false

## Two types of method

### 1)instance method

Any method declared inside the class is called instance method.

One instance method can call directly other instance method.

It must required any object to be called because it operates on the instance variables.

### 2)static method or class method

Any method declared inside the class with static keyword is called static method or class method.

It is not property of any object it is a property of class itself.

So it does not required any object to be called.

You can call static method by just it's name in the same class.

From outside the class you can use classname to call the static method

## Constructor

It is used to initialize the object that means to construct the object.

It is called automatically when an object or instance created.

It has the same name as the classname.

It can not have any return type

It can have zero or more parameters

A class can have more than one constructor but they must be different from

- 1)no of parameters or
- 2)type of parameters or
- 3)sequence of parameters

This concept is called constructor overloading

If you do not provide any constructor in class then java provides default constructor which has no parameters.

```
String s1=new String();---true
```

```
String s2=new String("jay");--true
```

```
String s3=new String(10)----false
```

```
Integer i1=new Integer(10);--true
```

```
Integer i1=new Integer("10");-true
```

### **C++---destructor**

### **Java---finalize method**

Garbage collection----(GC)----it is an automatic memory management techniques.

GC call the finalize method automatically before object destruction.

```
New classname();---unreferenced objects or unused object
```

### **Use of this keyword**

- 1)To refer the current object in class method or constructor.
- 2)To call one constructor from another constructor of same class but It must be the first statement.

### **Java ---It is not pure object oriented language**

Every thing must be represented as an object---pure

Primitive data types or values are not as an objects.

```
int a=10
```

```
float f=10.23f
```

```
char ch='4';
```

To represent primitive values as an objects java uses the concept of Wrapper class

Data types in java	wrapper class
--------------------	---------------

int	Integer
float	Float
char	Character
boolean	Boolean

**Boxing**----to convert primitive data types to object

```
int a=10;  
Integer i1=new Integer(a);
```

**Unboxing**-----to convert objects into primitive data types

```
int b=i1.intValue();
```

**Java.5.0,6.0.7.0**

**Autoboxing**

```
int a=10;  
Integer i1=10;  
Integer i1=a;
```

**Autounboxing**

```
int b=i1;
```

**Method Overloading or Function Overloading**

(static polymorphism or compile time or static binding or early binding)

Method name must be same(same class or parent class and child class)

But parameters must be different

- 1)No Of Parameters
- 2)Type Of Parameters
- 3)Sequence Of Parameters

**Return type may be of any type, access specifier may be of any *type*.**

**Variable no of Arguments or varargs**

```
void sum(int,int);  
void sum(int,int,int);  
void sum(int,int,int,int);
```

```
void sum(int... a)—varargs or variable no of arguments
{
}
```

It must be the last argument otherwise it will give compilation error.  
You can use only one parameter as varargs

void sum(int... a)-----You can pass any no of integers  
void sum(int[]... a)-----You can pass any no of integer arrays

### Enhanced For Loop or For Each Loop

It is specially designed to deal with array and collection.  
for(data type variablename : array name or collection name)  
{  
  
}

### Arrays in Java

It is collection of same types of data.

```
int a;  
a=45;
```

You can use array to store more than one values of same data types under single variable name.

### Declaration of Array

```
1)int a[]={10,20,30,40,50};  
    Or  
    int []a={10,20,30,40,50};
```

0	1	2	3	4
10	20	30	40	50



**2) by using new keyword**

```
int a[ ];  
a=new int[5];  
Or  
int a[]=new int[5];
```

0	1	2	3	4
45	67	56	89	23

```
a[0]=45;  
a[1]=67;  
a[2]=56;  
a[3]=89;  
a[4]=23;
```

**3) int a[]=new int[] {10,20,30,40,50};**

**Each array object has one property---length**

**Difference between 1) and 3) method**

```
int a[];  
a=new int[] {10,20,30,40,50}; valid
```

```
int a[ ]={10,20,30,40,50};  
int a[];  
a={10,20,30,40,50}---this is not valid
```

**Two Dimensional Array(Arrays of Array)(Rectangular Array)**

```
1)int a[][]={  
    {10,20,30},  
    {40,50,60},  
    {70,80,90}  
};
```

3\*3==9 values

Index	0	1	2
0	10	20	30
1	40	50	60
2	70	80	90

```
2)int a[][];
   a=new int[3][3];
```

First size----No of Arrays

Second size---No of elements in each array

Or

```
int a[][]=new int[3][3];
```

a[2][2]=67;

Index	0	1	2
0			
1			
2			67

```
3)int a[][]=new int[][] { {10,20,30},{40,50,60},{70,80,90} };
```

**Non Rectangular Array(Variable No Of Columns Array)(Zigzag Array or Jagged Array)**

```
int a[][]=new int[3][];
a[0]=new int[5];
a[1]=new int [3];
a[2]=new int[7];
```

--	--	--	--	--

--	--	--

		67			
--	--	----	--	--	--

a[2][3]=67;

## Array Of Objects

```

1)student s[]={new student(),new student(),new student()};
2)student s[];
   s=new student[3];
   Or
student s[]=new student[3];

   s[0]=new student();
   s[1]=new student();
   s[2]=new student();
3)student s[]=new student[] {new student(),new student(),new student()};
    
```

## Use of final Keyword

### 1)final variable

It becomes constant so you can not change the value of final variable.

Blank final variable—instance variable,static variable,local variable

You must use constructor or object initialize r block to initialize final instance variable.

You must use static block to initialize final static variable.

### 2)final method

It prevents method overriding.

you can not override final method.

parent class method-----**public final** void show()

child class-----can not override

parent class method-----**protected final** void show()

child class-----can not override

parent class method----- **final** void show()

child class-----can not override

parent class method-----**private final** void show()

child class-----can not override

**Child class can define it's own method with the same name and same signature.**

### 3)final class

It prevents inheritance.

You can not make subclass of final class.

## Pojo File

Plain old java object file(java Bean)(Entity class or persistence class or model)

Normal java file which contains private properties and public getter and setters of each property

Getters and Setters

Student-----rollno,name,address,phoneno

int getrollno()

```
{
    Return rollno;
}
```

void setrollno(int r)

```
{
    Rollno=r;
}
```

## Access Specifier or Visibility Mode

Member Access Specifire	In same Class	In same Package Child Class	In same Package but Different Class	Different Package in Child class	Different Package in Different Class
private	yes	no	no	no	no
Default No Modifire (Package private)	yes	yes	yes	no	no
protected	yes	yes	yes	yes	no
public	Yes	Yes	Yes	Yes	yes

## Inheritance

New Object = Existing Object + New Features

### Types of Inheritance

#### 1)single level

A----->B

A class—parent class or super class or base class

B class---child class or subclass or derived class

#### 2)Multilevel Inheritance

A----->B----->C----->D

D<-----A,B,C

C<-----A,B

B<-----A

#### 3)Hierarchical or Tree

A----->B

A----->C

B----->D

B----->E

C----->F

C----->G

#### 4)Multiple(Java does not support multiple inheritance instead of this java uses the concept of Interface)

A----->D

B----->D

C----->D

#### 5)Hybrid or Multipath or Diamond Shape

A----->B

A----->C

B----->D

C----->D

(Java does not support this type of Inheritance)

## Employee class

codeno

name

address

phoneno

## Manager

codeno

name

address

phoneno

**department**

## Engineer

codeno

name

address

phoneno

**qualification**

## Supervisor

codeno

name

address

phoneno

**no of hours**

## Manager

inherited common information

department

## Engineer

inherited common information

qualification

## Supervisor

inherited common information

noofhours

## Method Overriding or Method Hiding

A-----B

A-----public void show(int a)

B-----public void show(int b)

## Rules of Method Overriding

### 1)Access Specifier-----Must be same or wider

Access specifier----same or wider access specifier is allowed

parent class----method----default

child class—method-----default,protected,public

parent class----method----protected

child class—method-----protected,public

parent class----method----public

child class—method-----public

parent class----method---private

you can not override private method

child class—method-----private,default,protected,public

### 2)Return Type-----Must be same or subclass

Return type-----same or subclass

datatype-----same

classname-----same or subclass

emp show()-----parent

emp show()-----child

manager show()-----child

engineer show()-----child

supervisor show()---child

**3)Method Name-----must be same**

**4)Parameters-----must be same**

**5)Exception Rule**

**ExceptionHandling with MethodOverriding**

If the superclass method does not declare an exception, subclass overridden method cannot declare the checked exception but it can declare unchecked exception.

If the superclass method declares an exception, subclass overridden method can declare same, subclass exception or no exception but cannot declare parent exception.

**Dynamic Method Dispatch or Virtual Method Invocation**

A-----B

A-----C

A-----D

A a1;-----Reference of Parent Class

a1=new B();----Object of Child Class

a1=new C();

a1=new D();

**Dynamic Method Dispatch or Virtual Method Invocation**

Reference-----Parent Class

Object-----Child Class

emp e;

e=new manager();

behaviour mnager call

e=new engineer();



behaviour enginner call

```
e=new supervisor();
```

behaviour supervisor call

## Polymorphic Argument

```
void disp(Manager m1)
```

```
{
```

```
    m1.getdata();
```

```
    m1.showdata()
```

```
}
```

```
disp(new Manger());
```

```
void disp(Engineer e1)
```

```
{
```

```
    e1.getdata();
```

```
    e1.showdata()
```

```
}
```

```
disp(new Engineer());
```

```
void disp(emp e1)-----Polymorphic Argument
```

```
{
```

```
    e1.getdata();
```

```
    e1.showdata()
```

```
}
```

```
disp(new engineer());
```

```
disp(new manager());
```

```
disp(new supervisor());
```

## Abstract class

If a class contains at least one method abstract then class must be declared abstract.

Abstract class may contains abstract(method without body) as well as concrete(method with body) method.

It may have variables of any type with any access specifier.

It can have constructor

You can not create an object of an abstract class so you have to extends the abstract class.

The child class must override all the abstract method of an abstract class otherwise it must be declared an abstract.

## Interface

It contains only abstract method(by default all the methods of interface **are public and abstract**).

By default all the variables declared in the interface are public static and final.

You must implements an interface into child class.

Child class must override all the methods of interface otherwise it must be declared an abstract.

You can not create an object of interface but you can create reference variable of an interface.

```
interface name
{
    method declaration----public and abstract----by default
    variable declaration---public static final--by default
}
class classname implements interfacename
{
}
class classname implements interfacename1,interfacename2,interfacename3
```

```
{
}
class classname extends parentclass implements interface1,interface2,interface3
{
}
interface interfacename extends interfacename
{
}
interface interfacename extends interface1,interface2,interface3
{
}
```

## String

You can create String objects by two method

- 1)String Literal
- 2)String object by using new

### 1)String Literal

```
String s1="Jaysukh Patel";
String s2="Jaysukh Patel"
String s3="Jaysukh Patel";
String s4="jaysukh Patel";
```

if(s1==s2)-----both are same---compare the references  
if(s1.equals(s2))-----both are same---compare the contents

if(s1==s4)-----both are different  
if(s1.equals(s4))-----both are different

### 2)String Objects by using new

```
String s1=new String("jaysukh Patel");
String s2=new String("jaysukh Patel");
String s3=new String("jaysukh Patel");
String s4=new String("Jaysukh Patel");
```

if(s1==s2)-----both are different

if(s1.equals(s2))----Both are same

```
String s1="Jaysukh Patel";  
String s2=new String("Jaysukh Patel");
```

if(s1==s2)-----both are different  
if(s1.equals(s2))----Both are same

### Exception Handling

Errors----are of two types

- 1)Comiplation Error---Syntax error
- 2)Runtime Error----Abnormal Condition which causes the program termination.

**Exception---It is runtime abnormal condition which causes program termination.**

### Exception Handling

To detect the error.

Take Appropriate Action and Give message to user.

Program will continue.

### Five Keywords to handle the Exception

try  
catch  
throw  
throws  
finally

### Try - Catch

```
try  
{  
    //statment--error
```

```
}  
catch(argument e)  
{  
}
```

## Try With Multiple Catch

```
try  
{  
    //statement--error-multiple  
}  
catch(argument e)  
{  
}  
catch(argument e)  
{  
}
```

## finally

1)

```
try  
{  
}  
finally  
{  
}
```

2)

```
try
```

```
{  
}  
catch()  
{  
}  
finally  
{  
}
```

3)

```
try  
{  
}  
catch() {}  
catch {}  
finally {}
```

It must be executed whether exception occurs or not.

It is used to do the cleaning activity like to free the resources linked in the program, to close the file, to close any stream.

### Throw Keyword

You can explicitly throw any object of Throwable or subclass of Throwable.

throw Throwable or subclass of Throwable instance

It can be used to throw User defined exception or custom exception or pre defined Java Exception.

### Throws Keyword

It must be used after method definition.

void show() throws exception list (comma separated list)

```
{  
}
```

## Difference Between Array and ArrayList

### Array

Static Memory Allocation.

Collection of same type of values.

### ArrayList

It is a collection object.

Dynamic Memory Allocation.

Collection of different types of objects which are of same kind(parent).

Ordered Collection

Duplicate Object Allowed