Training on Java Notes

Java Introduction

• Author of Java	James Gosling
• Vendor	Sun Micro System(since merged into Oracle Corporation)
• Project name	Green Project
• Type	Open source & free software
• Initial Name	OAK language
• Present Name	Java
• Extensions	.java & .class & .jar
• Initial version	Jdk 1.0 (java development kit)
• Operating System	Multiple Operating System
• Implementation Lang	c, c++.....
• Symbol	Coffee cup with saucer
• Objective	Develop web Application
• Slogan	WORA(write once run anywhere)

Importance Of Core Java

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

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

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

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

4) Java is used to develop Mobile Applications.

5) Java is used to develop Embedded System.

6) Java is used to develop SmartCards.

7) Java is used to develop Robotics.

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

Java Features

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

Java Features In Details

1. Simple:-

Java is a simple programming language because:

-- Java technology has eliminated all the difficult and confusion oriented concepts like pointers, multiple inheritance in the java language.

-- The c, cpp syntaxes easy to understand and easy to write. Java maintains C and CPP syntax mainly hence java is simple language.

-- Java tech takes less time to compile and execute the program.

2. Object Oriented:-

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

Continue--

3. Platform Independent:-

-- Compile the Java program on one OS (operating system) that compiled file can execute in any OS (operating system) is called Platform Independent Nature.

-- The java is platform independent language. The java applications allow its applications compilation one operating system that compiled (.class) files can be executed in any operating system.

4. Architectural Neutral:-

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

5. Portable:-

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

Continue--

6. Robust:-

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

1. Exception Handling

2. Memory Allocation

JAVA is Robust because

-- JAVA is having very good predefined Exception Handling mechanism whenever we are getting exception we are having meaning full information.

-- JAVA is having very good memory management system that is Dynamic Memory (at runtime the memory is allocated) Allocation which allocates and deallocates memory for objects at runtime.

7. Secure:-

-- To provide implicit security Java provide one component inside JVM called Security Manager.

-- To provide explicit security for the Java applications we are having very good predefined library in the form of java.Security.package.

Continue--

8. Dynamic:-

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

9. Distributed:-

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

-- Standalone applications are java applications it doesn't need client server architecture.

-- Web applications are java applications it need client server architecture.

-- Distributed applications are the applications the project code is distributed in multiple number of jvm's.

10. Multithreaded:-

-- Thread is a light weight process and a small task in large program.

-- If any tech allows executing single thread at a time such type of technologies is called single threaded technology.

-- If any technology allows creating and executing more than one thread called as multithreaded technology called JAVA.

Continue--

11. Interpretive:-

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

12. High Performance:-

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

Install Software And Set The Path

1) Download the software.

2) Install the software in your machine.

3) Set the environmental variable.

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

To download the software open the fallowing web site.

http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

for 32-bit operating system please click on Windows x86 :- 32- bit operating system

for 64-bit operating system please click on Windows x64 :- 64-bit operating system

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

Local Disk c: ----program Files------java--jdk(java development kit), jre(java runtime environment)

To check whether the java is installed in your system or not go to the command prompt. To open the command prompt Start --------run----open: cmd---ok Command prompt is opened.

Continue--

In the command prompt type :- javac

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

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

Whenever we are typing javac command on the command prompt

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

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

To set the environmental variable:-

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

Paste the copied path then click on OK.

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

Write First Java Program

Steps to write first java program:-

Step-1:- Select Editor.

Step-2:- Write the application.

Step-3:- save the application.

Step-4:- Compilation Process.

Step-5:- Execution process.

Step1:- Select Editor

Editor is a tool or software it will provide very good environment to develop java application. Ex :

Notepad, Notepad++,edit Plus.....etc

Continue--

Step 2:- Write a program.

-- Write the java program based on the java API(Application Programming Interface) rule and regulations .

-- Java is a case Sensitive Language so while writing the program you must take care about the case (Alphabet symbols).

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

Continue--

Step3:- Save the application.

-- After writing the application must save the application by using (.java) extension.

-- While saving the application must fallow two rules :-

1. If the source file contains public class then public class and the name and source file must be same publicClassName.java).

Otherwise compiler generate error message.

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

Step-4:- Compilation process.

Compile the java application by using javac command.

Syntax:-

javac Test.java

Step-5:- Execution process.

Run /execute the java application by using java command.

Syntax:-

java class-name

java Test

Hello World

Dynamic Input In Java

Java.util.Scanner(Dynamic Input):-

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

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

Scanner s = new Scanner(System.in);

To get int value ----> s.nextInt()

To get float value ---> s.nextFloat()

To get byte value ---> s.nextbyte()

To get String value ---> s.next()

To get single line ---> s.nextLine()

To close the input stream ---> s.close()

Example Application - 1

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

Continue--

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

```
enter emp no 1001
enter emp name Ravi
enter emp salary 12000.0
emp no----->1001
emp name------>Ravi
emp salary------->12000.0
```

Example Application – 2

```
//WAP to find volume and surface area of cuboid
import java.util.Scanner;
class Cuboid
{
public static void main(String [] args)
{
int l,b,h;
Scanner sc=new Scanner(System.in);
System.out.print(“Enter length of cuboid : ”);
l=sc.nextInt();
System.out.print(“Enter breadth of cuboid : ”);
b=sc.nextInt();
System.out.print(“Enter height of cuboid : ”);
h=sc.nextInt();
```

Continue--

```
int v=l*b*h;
int sa=2*(l*b+b*h+h*l);
System.out.println("Volume of cuboid : "+v);
System.out.println("Surface Area of cuboid : "+sa);
}
}
```

Output:-

```
Enter length of cuboid: 10
Enter breadth of cuboid : 5
Enter height of cuboid : 4
Volume of cuboid : 200
Surface Area of cuboid : 220
```

Decision Control (If)

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

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

if Statement:- if is a keyword which works like decision control. We attach a condition with if statement.

If given condition is true then code will executed and if given condition is false then it do nothing.

Syntax:-

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

Example Application -1

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

Decision Control (If- Else)

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

Syntax of if –else:-

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

Example Application -2

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

Example Application -3

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

Output

O/P 1:-

Enter the value of a : 1

Enter the value of b : 2

Enter the value of c : 3

Roots are imaginary

O/P 2:-

Enter the value of a : 1

Enter the value of b : -2

Enter the value of c : 1

Root 1=1.0

Root 2=1.0

Decision Control (Ladder If - Else)

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

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

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

Example Application - 4

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

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

```
import java.util.Scanner;
class ElectricityBill
{
public static void main(String [] args)
{
int unit;
double bill=0.0;
Scanner sc=new Scanner(System.in);
System.out.print("Enter the no. of units consumed : ");
unit=sc.nextInt();
```

Continue--

```
if(unit<=150)
{
bill=unit*2.40;
}
else if(unit>150 && unit<=300)
{
bill=(150*2.40)+(unit-150)*3.00;
}
else
{
bill=(150*2.40)+(150*3.00)+(unit-300)*3.20;
}
System.out.println("Your bill="+bill);
}
}
```

O/P:-

Enter the no. of units consumed : 200
Your bill=510.00

Decision Control (switch)

Switch is a keyword in java which works as case control. It is used to make menu based program.

1) Switch statement is used to declare multiple selections.

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

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

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

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

Syntax Of Switch

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

Example Application - 5

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

Continue--

```
case 2:
System.out.print(“Enter temperature in f : ”);
f=sc.nextDouble();
c=(f-32)*5/9;
System.out.println(“Temperature in c=”+c);
break;
default:
System.out.println(“Invalid choice”);
break;
}
}
}
```

O/P:-

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

Loop Controls In Java

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

1. while

2. for

3. do – while

4. for each

While Loop

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

```
Initialization of loop counter;

while(Condition)

{

//Body of Loop

Updation of loop counter;

}
```

Example Application -1

```
// Develop a program in java to generate series of even numbers from 1- 100
class Test {
public static void main(String [] args) {
int i=1;
while(i<=100)
{
if(i%2==0) {
System.out.print(i+" ");
}
i++;
}
}
}
```

Example Application -2

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

Example Application -3

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

O/P:- Enter the number to find factorial : 5
Factorial = 120

For Loop

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

Syntax of for loop:-

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

Example Application - 4

```
//Develop a program in java to generate Fibonacci sequence
import java.util.Scanner;
class Test {
public static void main(String [] args) {
int n1=0,n2=1,n3,n,i;
Scanner sc=new Scanner(System.in);
System.out.print(“How many terms ? ”);
n=sc.nextInt();
System.out.println(“Fibonacci Sequence”);
System.out.println(n1);
System.out.println(n2);
for(i=1;i<=n-2;i++) {
n3=n1+n2;
System.out.println(n3);
n1=n2;
n2=n3;
}
}
}
```

Nested For - Loop

If you use a for loop inside another for loop then it is called as nested for loop.

Syntax of Nested for – loop:-

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

Example Application - 5

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

Do – While Loop

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

Syntax of do-while loop:-

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

Example Application - 6

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

Array In Java

-- Arrays are used to represent group of elements as a single entity but these elements are homogeneous & fixed size.

-- The size of Array is fixed it means once we created Array it is not possible to increase and decrease the size.

-- Array in java is index based first element of the array stored at 0 index.

Advantages of array:-

-- Instead of declaring individual variables we can declare group of elements by using array it reduces length of the code.

-- We can store the group of objects easily & we are able to retrieve the data easily.

-- We can access the random elements present in the any location based on index.

-- Array is able to hold reference variables of other types.

Continue--

Different ways to declare an array:-

int[] values;

int []values;

int values[];

Declaration& instantiation & initialization :-

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

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

a[0]=10; //initialization

a[1]=20;

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

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

a[99]=40;

Example Application -1

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

String Class In Java

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

Import Built-in Methods Of String Class:-

toUpperCase():- The toUpperCase() method of String class converts string into the Upper Case.

toLowerCase():- The toLowerCase() method of String class converts string into the Lower Case.

length():- The length() method of String class find the no. of characters in given string.

equals():- The equals() method of String class check the given two strings are equal or not. This method returns boolean value.

equalsIgnoreCase():- The equalsIgnoreCase() method of String class also check the given two strings are equal or not. This method also return boolean value but this method avoid case sensitivity.

trim():- The trim() function of String class removes starting and ending spaces of given string.

Continue--

contains():- The contains() method check the given char is presented in string or not.

replace():- The replace() method of String class is used to replace() one character to another.

charAt():- The charAt() method of String class is used to find character at given position.

indexOf():- The indexOf() method find the position of given character.

split():- The split() method return a split string matching reqex.

valueOf():- The valueOf() method converts given type into String. It is an overloaded method.

isEmpty():- The isEmpty() method checks if string is empty.

Example Application -2

```
//Develop a program in java to compare two strings for equality
import java.util.Scanner;
class Test {
public static void main(String [] args) {
Scanner sc=new Scanner(System.in);
System.out.print(“Enter first string : ”);
String str1=sc.nextLine();
System.out.print(“Enter second string : ”);
String str2=sc.nextLine();
if(str1.equals(str2)==true)
System.out.println(“Both strings are equal”);
else
System.out.println(“Both strings are not equal”);
}
}
```

Example Application -3

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

Example Application - 4

```
//Develop a program in java to check given string is palindrome or not.
import java.util.Scanner;
class Test {
public static void main(String [] args) {
Scanner sc=new Scanner(System.in);
System.out.print(“Enter a string : ”);
String str=sc.nextLine();
String revstr=””;
for(int i=0;i<str.length();i++) {
revstr=revstr+str.charAt(i)+””;
}
if(str.equals(revstr)==true)
System.out.println(“String is palindrome”);
else
System.out.println(“String is non-palindrome”);
} }
```

Example Application - 5

```
/*Develop a program in java to take a sentence as input. Find a word in sentence. Replace the word with
another word in sentence.*/
import java.util.Scanner;
class Test {
public static void main(String [] args){
String sentence, fw, rw;
Scanner sc=new Scanner(System.in);
System.out.print(“Enter a sentence : ”);
sentence=sc.nextLine();
System.out.print(“Find what : ”);
fw=sc.nextLine();
System.out.print(“Replace with : ”);
rw=sc.nextLine();
System.out.println(“Modified sentence : ”+sentence.replace(fw,rw));
}
}
```

Example Application - 6

```
/* Develop a program in java to take the user name as input and display the short name.
E.g. I/P: Ajay Kumar Singh
O/P: A.K.Singh*/
import java.util.Scanner;
class Test{
public static void main(String [] args){
Scanner sc=new Scanner(System.in);
System.out.print("Enter your name : ");
String name=sc.nextLine();
String shortname[]=name.spilt(" ");
System.out.print("Your short name : ");
for(int i=0;i<shortname.length-1;i++)
{
System.out.print(shortname[i]+".");
}
System.out.print(shortname[shortname.length-1]);
}
}
```

Method In Java

-- Methods are used to write the business logics of the project.

-- Coding conversion of method is method name starts with lower case letter if method contains more than One word then every inner word starts with uppercase letter. Example:- post() , charAt() , toUpperCase() , compareToIgnoreCase().....etc

-- There are two types of methods **1. Instance method 2. Static method**

-- Inside the class it is possible to declare any number of instance methods & static methods based on the developer requirement.

-- It will improve the reusability of the code and we can optimize the code.

Continue--

Instance Method:- The instance method is also called non-static method. These methods are declared without using static modifier. These methods can call by using the object of class. Without using object we can't call instance method.

Static Method:- The static methods are declared by using static keyword. These method are also called class methods. There is no need of object to call static methods

Syntax of declaration of method in Java:-

```
<modifier> <return_type> method_name(parameters)
{
//code
}
```

E.g.

```
static int add(int x, int y)
{
return (a+b);
}
```

Example Application -1

```
//Find the volume of cuboid using method
import java.util.*;
class MethodDemo1 {
public static void main(String [] args){
int l,b,h,v;
Scanner sc=new Scanner(System.in);
System.out.println("Enter length, breadth and height of cuboid");
l=sc.nextInt();
b=sc.nextInt();
h=sc.nextInt();
v=volume(l,b,h);
System.out.println("Volume of cuboid="+v);
}
static int volume(int x,int y,int z){
return (x*y*z);
} }
```

Recursion

When a function call itself then it is called ’Recursion ’. For example To find the factorial of given number:-

5!=5*4*3*2*1

5!=5*4!

.

.

n!=n*(n-1)!

fact(n)=n*fact(n-1);

Example Application - 2

```
//Develop a program to find factorial of given number using ‘Recursion’.
import java.util.*;
class Test {
static long fact(int n) {
if (n==0 || n==1)
return 1;
else
return n*fact(n-1);
}
public static void main(String [] args){
Scanner sc=new Scanner(System.in);
System.out.print(“Enter the number to find factorial : ”);
int n=sc.nextInt();
System.out.println(“Factorial=”+fact(n));
}
}
O/P:- Enter the number to find factorial : 5
Factorial=120
```

Object Oriented Programming System

Continue--

The Object Oriented Programming System is a mechanism of Software Development. It have four concepts:-

1. Abstraction

2. Encapsulation

3. Inheritance

4. Polymorphism

Any Programming Language which follows these concepts is called Object Oriented Programming Language.

Abstraction In Java

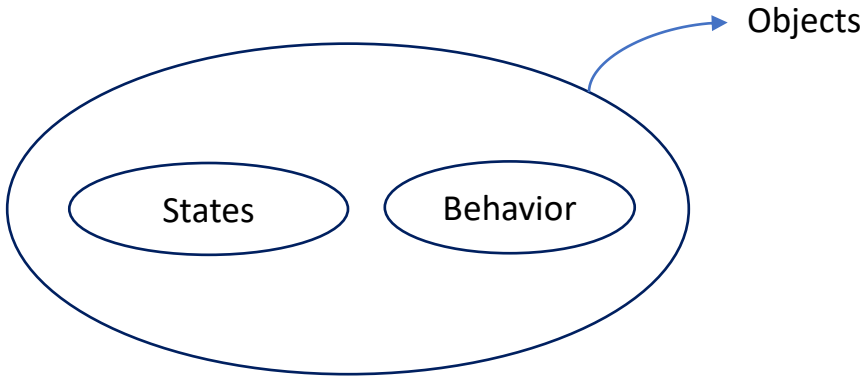
Abstraction is selecting data from a larger pool to show only the relevant details to the object. It helps to reduce programming complexity and effort. In Java, abstraction is accomplished using Abstract classes and interfaces. It is one of the most important concepts of OOPs. Abstraction in Java can be achieved using Abstract Class and Abstract Method.

Abstract Class:- A class which is declared “abstract” is called as an abstract class. It can have abstract methods as well as concrete methods. A normal class cannot have abstract methods.

Abstract Methods:- A method without a body is known as an Abstract Method. It must be declared in an abstract class. The abstract method will never be final because the abstract class must implement all the abstract methods.

Encapsulation In Java

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Other way to think about encapsulation is, it is a protective shield that prevents the data from being accessed by the code outside this shield.

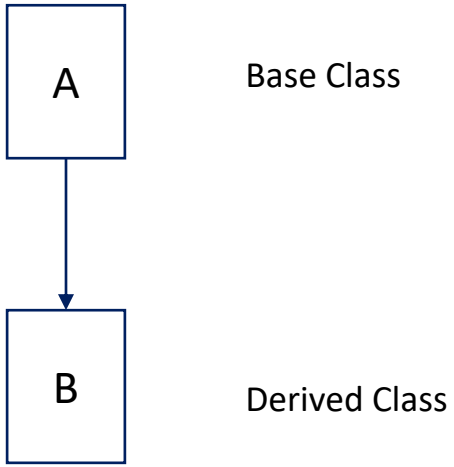


Inheritance In Java

The Inheritance is a very nice feature of Object oriented programming . In Inheritance we can create a new class by using existing class. The existing class is called base class and new created class is called derived class. The concept of Inheritance is also called “Reusability”.

.

.



Polymorphism In Java

The term “Polymorphism” means one thing many forms. There are two types of Polymorphism.

Compile Time Polymorphism:- The compile time polymorphism contains the concept of overloading. In java there is a concept of method overloading. In method overloading the method name is same but parameters are different. Based on method parameters it is decided at compilation time that which method call from where.

Run Time Polymorphism:- The run time polymorphism contains the concept of overriding. In java there is a concept of method overriding. The re-writing of base class method to derived class is called ‘Method Overriding’.

.

.

Concept Of Class

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

Declaration of Class:-

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

Example Application -3

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

Example Application -4

```
//Use of private access specifier
class Car
{
private String make; //private data member
private String color;//private data member
private int price; //private data member
public void setCar (String make, String color, int price)
{
this.make=make;
this.color=color;
this.price=price;
}
public void display()
{
System.out.println(“Car make-> ”+this.make);
System.out.println(“Car color-> ”+this.color);
System.out.println(“Car price-> ”+this.price);
}
};
.
```

Continue--

```
class Test
{
public static void main(String [] args)
{
Car c=new Car();
c.setCar(“Tata”,”Silver”,650000);
c.display();
}
}
```

O/P:-

```
Car make->Tata
Car color->Silver
Car Price->650000
```

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

.

.

Concept Of Constructor

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

Rules to declare constructor:-

1) Constructor name class name must be same.
2) Constructor is able to take parameters.
3) Constructor not allowed explicit return type (return type declaration not possible).

There are two types of constructors:-

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

.
.

Default Constructor

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

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

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

.
.

Application Before Compilation

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

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

.

.

Application After Compilation

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

Example Application - 5

```
class Employee {
//instance variables
int eid;
String ename;
double esal;
Employee(int eid,String ename,double esal) //local variables
{
//conversion (passing local values to instance values)
this.eid = eid;
this.ename = ename;
this.esal = esal;
}
void display() {
//printing instance variables values
System.out.println("****Employee details****");
System.out.println("Employee name :-->"+ename);
System.out.println("Employee eid :-->"+eid);
System.out.println("Employee esal :-->"+esal);
}
.
.
```

Continue--

```
public static void main(String[] args)
{
// during object creation parameterized constructor executed
Employee e1 = new Employee(111,Krishn",60000);
e1.display();
Employee e2 = new Employee(222,"Rohit",70000);
e2.display();
Employee e3 = new Employee(333,"Yashi",80000);
e3.display();
}
}
```

Output

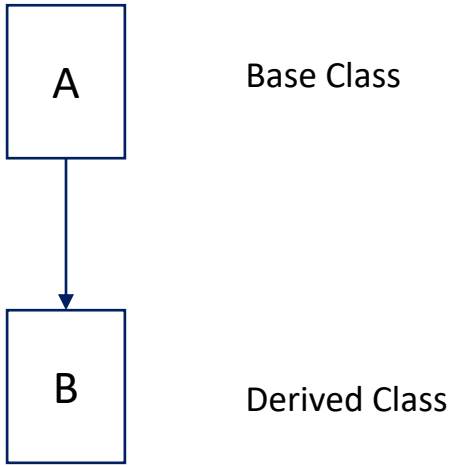
```
****Employee details****
Employee name :-->Krishn
Employee eid :-->111
Employee esal :-->60000.0
****Employee details****
Employee name :-->Rohit
Employee eid :-->222
Employee esal :-->70000.0
****Employee details****
Employee name :-->Yashi
Employee eid :-->333
Employee esal :-->80000.0
```

Inheritance In Java

The Inheritance is a very nice feature of Object oriented programming . In Inheritance we can create a new class by using existing class. The existing class is called base class and new created class is called derived class. The concept of Inheritance is also called “Reusability”.

.

.



Syntax Of Inheritance In Java

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

Types Of Inheritance In Java

There are following types of inheritance are supported in java:-

1. Single/ Simple Inheritance

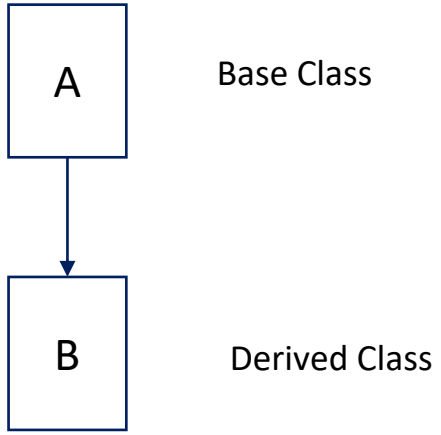
2. Hierarchical Inheritance

3. Multi – level Inheritance

4. Hybrid Inheritance

Single/ Simple Inheritance

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



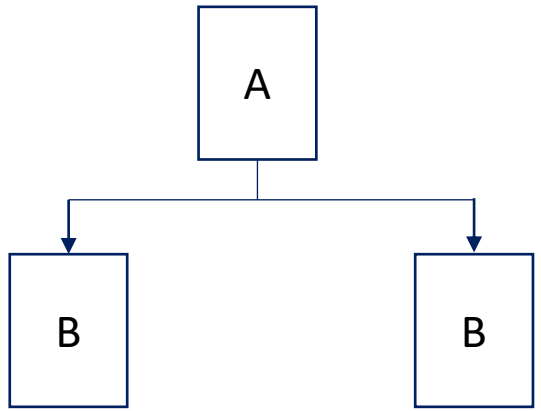
Base Class

Derived Class

```
class A
{
.....;
.....;
}
class B extends A
{
.....;
.....;
}
```

Hierarchical Inheritance

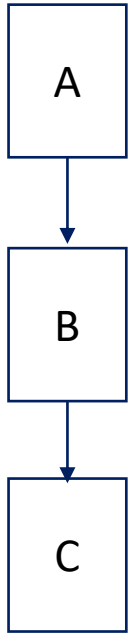
In Hierarchical Inheritance there is a single base class and multiple derived class.



Hierarchical Inheritance

```
class A
{
.....;
.....;
}
class B extends A
{
.....;
.....;
}
class C extends A
{
.....;
.....;
}
```

Multi-level Inheritance

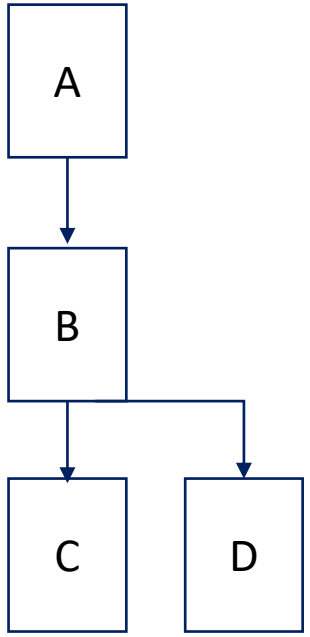


Multi-level Inheritance

```
class A
{
.....;
.....;
}
class B extends A
{
.....;
.....;
}
class C extends B
{
.....;
.....;
}
```

Hybrid Inheritance

If we combine more than one inheritance then resultant inheritance is called Hybrid Inheritance. Since Multiple Inheritance is not supported in java, so we can not include multiple inheritance to make Hybrid Inheritance.



Hybrid Inheritance

```
class A {
.....;
}
class B extends A {
.....;
}
class C extends B {
.....;
}
class D extends B {
.....;
}
```

Example Application - 1

/*Develop a program in java to create a class Rundog. In Rundog class make a method bark(), in bark() method display the rundog name and voice. By extending Rundog class create a new class named Bulldog. In Bulldog class make a method grawl(), in grawl() method display bulldog name and voice.*/

```
class Rundog
{
public void bark()
{
System.out.println(“Tommy.....”);
System.out.println(“Bho.....Bho.....”);
}
}
class Bulldog extends Rundog
{
public void grawl()
{
System.out.println(“Tuffy.....”);
System.out.println(“Gurr.....Gurr.....”);
}
}
```

Continue--

```
class Test
{
public static void main()
{
Bulldog dog=new Bulldog();
dog.bark();
dog.grawl();
}
}
```

Example Application – 2

```
import java.util.Scanner;
class Shape {
protected int s; //protected data member
public void setValue(int x) //public method to initialize data member
{
s=x;
}
}
class Cube extends Shape {
public int volume() {
return (s*s*s);
}
}
class Square extends Shape {
public int area() {
return(s*s);
}
}
```

Continue--

```
class Test
{
public static void main(String [] args)
{
Scanner sc=new Scanner(System.in);
int x;
System.out.print(“Enter side of cube : ”);
x=sc.nextInt();
Cube cu=new Cube();
cu.setValue(x);
System.out.println(“Volume of cube : ”+cu.volume());
System.out.print(“Enter side of square : ”);
x=sc.nextInt();
Square sq=new Square();
sq.setValue(x);
System.out.println(“Area of square : ”+sq.area());
}
}
```

Polymorphism In Java

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

1. Compile Time Polymorphism (Overloading)

2. Run Time Polymorphism (Overriding)

Compile time polymorphism [Overloading]:-

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

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

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

```
void m1(int a){}

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

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

```
void m1(int a){}

void m1(char ch){}
```

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

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

Method Overloading

In java programming language you can give same name to multiple methods but their arguments should be different. Based on method arguments it is decided at compilation time that which method call from where. It is called method overloading.

Example Application - 1

/*Create a class with names Shape in Shape class make three method with same name area (method overloading). First method find the area of square, second method find the area of circle and third method find the area of rectangle. Now test the class Shape.*/

```
import java.util.*;
class Shape {
public int area(int s) {
return (s*s);
}
public int area(int l,int b) {
return (l*b);
}
public double area(float r) {
return (3.14*r*r);
} }
```

Continue--

```
class Test {
public static void main(String [] args) {
int s,l,b,a1,a2;
double a3;
float r;
Scanner sc=new Scanner(System.in);
Shape sh=new Shape();
System.out.print(“Enter side of square : ”);
s=sc.nextInt();
System.out.print(“Enter length and breadth of rectangle : ”);
l=sc.nextInt();
b=sc.nextInt();
System.out.print(“Enter radius of circle : ”);
r=sc.nextFloat();
a1=sh.area(s);
a2=sh.area(l,b);
a3=sh.area( r);
System.out.println(“Area of square=”+a1);
System.out.println(“Area of rectangle=”+a2);
System.out.println(“Area of circle=”+a3);
}}
```

Example Application – 2

```
/* Constructor Overloading:-The class contains more than one constructors with same name but different arguments is called constructor overloading. */
class Test {
//overloaded constructors
Test() {
System.out.println("0-arg constructor");
}
Test(int i) {
System.out.println("int argument constructor");
}
Test(char ch,int i) {
System.out.println(ch+"-----"+i);
}
public static void main(String[] args) {
Test t1=new Test(); //zero argument constructor executed.
Test t2=new Test(10); // one argument constructor executed.
Test t3=new Test('a',100);//two argument constructor executed.
}
}
```

Method Overriding

The re-writing of base class method to derived class is called method overriding.

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

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

a. If child is satisfied use parent class method.

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

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

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

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

Rules For Method Overriding

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

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

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

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

5) While overriding check the covariant-return types.

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

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

Example Application - 3

```
// Develop a program to demonstrate concept of method overriding.
class A {
public void m1() {
System.out.println(“m1 of A”);
}
public void m2() {
System.out.println(“m2 of A”);
}
}
class B extends A {
public void m2() {
System.out.println(“m2 of B”);
}
public void m3() {
System.out.println(“m3 of B”);
}
}
```

Continue--

```
class Test
{
public static void main(String [] args)
{
A a1=new A();
a1.m1(); //m1 of A
a1.m2(); //m2 of A
B b1=new B();
b1.m1();//m1 of A
b1.m2();//m2 of B
b1.m3();//m3 of B
A a2=new B(); //Up-casting
a2.m1(); //m1 of A
a2.m2(); //m2 of B
}
}
```

Difference Between Method Overloading & Overriding

Method Overloading	Method Overriding
Method overloading is used to increase the readability of the program.	Method overriding is used to provide the specific implementation of the method that is already provided by its super class.
Method overloading is performed within class.	Method overriding occurs in two classes that have IS-A (inheritance) relationship.
In case of method overloading, parameter must be different.	In case of method overriding, parameter must be same.
Method overloading is the example of compile time polymorphism.	Method overriding is the example of run time polymorphism.
In java, method overloading can't be performed by changing return type of the method only. Return type can be same or different in method overloading. But you must have to change the parameter.	Return type must be same or covariant in method overriding.

Exception

The Dictionary meaning of exception is abnormal termination, when exception is occurred then Program is terminated abnormally and rest of code is not executed.

In java There are three types of exceptions .

---

1. Checked Exception
2. Unchecked Exception
3. Errors

Checked Exception

Checked Exception are those exception which are identified by compiler.

- ClassNotFoundException
- InterruptedException
- FileNotFoundException
- IOException
- SQLException etc.

Example Application - 1

```
package dbpack;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class DbManager
{
   Connection con=null;
   PreparedStatement ps=null;
   ResultSet rs=null;

   public DbManager()
   {
   try
   {
   Class.forName("com.mysql.jdbc.Driver");
   con=DriverManager.getConnection("jdbc:mysql://localhost:3306/oe?characterEncoding=utf-8","root","");
   }
```

Continue--

```
catch(ClassNotFoundException e1)
  {
  e1.printStackTrace();
  }
  catch(SQLException e2)
  {
  e2.printStackTrace();
  }
  }
  public boolean insertUpdateDelete(String query)
  {
  try
  {
  ps=con.prepareStatement(query);
  if(ps.executeUpdate()>0)
  {
  return true;
  }
  else
  {
  return false;
  }
  }
```

Continue--

```
catch(SQLException e1)
  {
  return false;
  }
  }
 public ResultSet select(String query)
 {
 try
 {
 ps=con.prepareStatement(query);
 rs=ps.executeQuery();
 }
 catch(SQLException e)
 {
 e.printStackTrace();
 }
 return rs;
 }
}
```

Unchecked Exception

Unchecked Exception which Exception are occurred at runtime.

- InputMismatchException
- ArrayIndexOutOfBounceException
- NullPointerException
- ArithmeticException , etc.

Error

Errors are Unchecked Exception, errors are occurred due to lack of System Resources.

Example Application - 1

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

public class bbbbb {
   public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);
      try
      {
         System.out.print("Enter first number = ");
      int a=sc.nextInt();
      System.out.print("Enter Second number = ");
   int b=sc.nextInt();
   System.out.println(a/b);
      }
      catch(ArithmeticException e)
      {
         System.out.println("Divisible by Zero is not Possible ");
      }
      catch(InputMismatchException e1)
      {
         System.out.println("only Interger Value Allowed");
      }
   }
}
```

Example Application - 2

```
import java.util.*;
class Ab
{
   public static void main(String [] args)
   {
      Scanner sc=new Scanner(System.in);
      try
      {
System.out.print("Enter first number = ");
int a=sc.nextInt();
System.out.print("Enter Second number = ");
int b=sc.nextInt();
System.out.println(a+b);
      }
      catch(InputMismatchException e)
      {
         System.out.println("Only Interger Value Allowed");
      }
}
}
```

Exception Handling

Exception Handling is a mechanism to handle exception to achieve normal execution program.

for exception handling in java we use try , catch , Finally , throw and throws Keywords.

In java you can handle Exception in two ways:-

i. By using try-catch blocks.
ii. By using throws Keyword.

Example Application - 1

1. **Exception handling by using try-catch blocks:-**

```
try
{
// Code which you want to
protect
}
catch (Exception Type variable )
{
// Code which is used to handle
//Exception
}
Finally
{
// Code which you want to
execute
// always
}
```

Example Application - 1

1. **Exception handling by using throws Keyword:-**

```
import java.io.*;
 class M
{
 void method()throws IOException
{
System.out.println("device operation performed");
 }
 }
 class Testthrows3
{
public static void main(String args[])throws IOException{//declare exception
      M m=new M();
       m.method();
    System.out.println("normal flow...");
 }
 }
```

Package

Package is a contains of classes , interfaces , and Subpackages.
Package is created by using package Keyword followed by package-name.

Example Application - 1

```
package mypack;
public class Myutil
{
public int add(int a , int b)
{
Return(a+b);
}
public int greatest (int a , int b)
{
int g=(a>b?a:y;)
return(g);
}
}
```

Command to automatic create folder name same as package name.

(Javac –d . Myutil.java)

Implemented Method

Implemented method means method is declared and also body Structure included mean Code is declare.

Example:-

```
void sayHello();
{

System.out.print(“ Hellow World!”);

}
```

Abstract Method

Abstract method means method is declared only.

Example:-

```
void  sayHellow();
```

Interface

Interface is the Collection of abstract method. It is used to achieve full abstraction. Interface is created by Interface_name. Inside body of Interface we declare abstrace method.

Example:-

```
Interface  interface1
{
  void m1();
  void m2();
}class TestInterface Implements interface1
{
public void m1()
{
System.out.print(“Hellow World!”);
}
public void m2()
{
System.out.print(“Hellow World!”);
}
Public static void main(String[]args)
{ TestInterface ti=new TestInterface();
ti.m1();
ti.m2();
}
}
```

Interface, Abstract class, class

Interface

If you have requirements but you don’t Know about its implementations ,
then you can use interface because interface contain abstract method only

- An interface Can extends another interface .
- We can’t create object of interface.

Abstract class

Abstract class is a class which Contain abstract method and Implemented method both.
If you have requirement, you know implementations. Then you can use Abstract class.

- An abstract class can implements interface.
- An abstract class can extends another abstract class.
- You can’t Create object of abstract class .

class

Class is the collection of Implemented methods . If you have requirements and you Can use class. because class Contains implemented method only.

- class can implements interface .
- class can extends abstract class .
- class can extends another class .
- You can Create object of class .