Object Oriented Programming in JAVA

L-1

@ Importance of Java

→ C/C++ are not portable & are not platform independent

Emergence of www demanded portable programs

∴ portability & security necessitated the invention of Java

@ Why are C/C++ not secure?

Because of pointers

* Java doesn't use pointers

@ Kinds of applications

→ Window based - That can run on window

→ Console based - That can run on command prompt

→ Web based - That can run on browser

@ JAVA Editions :-

→ J2SE → Java 2 standard Edition. To develop client-side standalone applications or applets

→ used for console based & window based applications

→ Most Important

→ J2ME → Java 2 Micro Edition
→ To develop applications for mobile devices such as cell phones.

→ J2EE → Java 2 Enterprise Edition
→ To develop server-side applications such as Java server pages.
→ Used for web based applications.

Q What is JAVA?
→ A general purpose object oriented language.
→ WORA (Write Once Run Anywhere) [Platform Independent]
→ Designed for easy web/internet applications.
→ Widespread acceptance

Q Difference b/w C & Java

→ C is a structure oriented, Java is object oriented & has mechanism to define classes object.

→ Java doesn't support an explicit pointer type
→ Java doesn't have preprocessor can't use #define, #include & #ifdef statements.

→ It doesn't include structures, unions, & enum datatype

→ doesn't include keywords like goto, sizeof & typedef

→ Adds labeled Break and Continue statements
→ Adds many features required for OOP

⑦ Diff b/w C++ & Java

→ doesn't support pointers to avoid unauthorized access of memory locations

→ doesn't include structure, union & enum datatypes

→ doesn't support operator overloading.

→ No preprocessor

→ doesn't perform automatic type conversion.

→ doesn't support global variables

→ doesn't allow default arguments

→ doesn't support inheritance of multiple super classes by sub class. This is accomplished by using Interference concept.

→ Cannot declare unsigned integers

→ Objects are passed by reference only.

⑧ New features of Java

→ Multithreading :- allows 2 or more pieces of the same programs to execute concurrently.

→ C++ has library functions that use a common header file. But Java replaces it with its own set of API classes

→ Adds packages & Interfaces.

→ Supports automatic garbage collection

→ break & continue statements have been enhanced in Java to accept labels as targets.

→ Use of unicode characters ensures portability.

Q Diff features from C++

→ C++ & Java supports boolean datatypes, C++ takes any non-zero value as true & zero as false.

True & false in Java are predefined literals that are values for a boolean expression.

→ Java has replaced the destructor function with finalize() function.

→ C++ supports exceptions handling similar to java.

L-2

__/__/__

⑤ Characteristics of JAVA

- Simple
- Object oriented
- Distributed
- Interpreted
- Robust
- Architecture neutral
- Portable
- Performance
- multi-threaded
- Dynamic.
- Secure

⑥ JAVA Environment

→ Java includes many development tools • classes & methods

- Development tools. — Java Development Kit (JDK)
- Classes & Methods - Java Standard Library (JSL) / Application Programming Interface (API)

→ JDC → Java compiles, Java Interpreter & many tools

→ API → Classes & Methods grouped into packages acc to their functionality.

JRE - Java Runtime Engine
JVM - Java Virtual Machine.

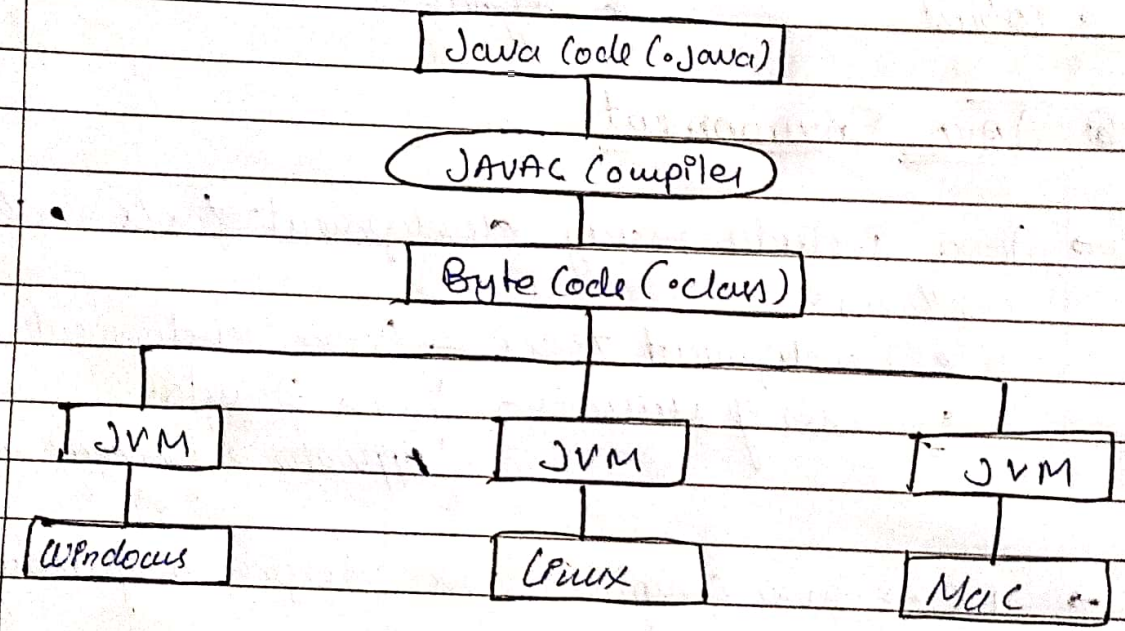
⑦ Working → Code —compile→ Intermediary file —Interpret→ Exe file

(Byte code / Class file / Encrypted program)

Byte code doesn't require support of any system
Exe file is independent of system

→ Byte code ~~doesn't~~ is given not exe file
→ Java is platform independent because exe is generated on client computers as only byte code is shared.

⑥ JAVA PROGRAM EXECUTION



L-3

First Program

```
class Example
{
    PSVM (String[] args)
    {
        System.out.println("Hello World");
    }
}
```

→ Public Static Void Main

* Save file as classname.java
* Java is case sensitive

L-4

JDK

→ Java development kit (JDK) is a software development environment used for developing Java applications & applets.

→ It includes JRE (Java Runtime Environment), an interpreter/loader (Java), a compiler (Javac), an archiver (Jar), a documentation generator (Javadoc) & other tools needed in Java development.

JRE

→ Java runtime Environment or Java JRE

→ JRE provides minimum requirements for executing a Java application.

→ It consists of JVM, core classes, & supporting files.

JVM is →

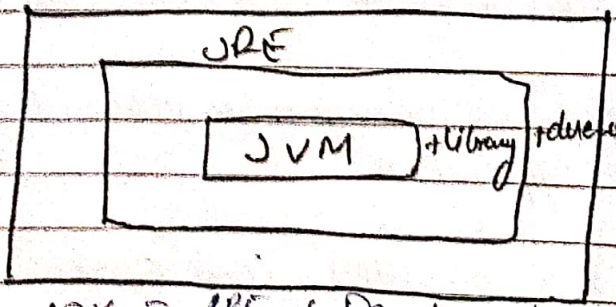
- A specification where working of Java virtual machine is specified
But implementation provider is independent to choose the algorithm. Its implementation has been provided by Sun & other companies.
- An implementation is a computer program that meets the requirements of the JVM specifications.
- Runtime Instance → whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

★ JDK Environment to develop & execute (run) the java program

Includes → Development tools
→ JRE

JDK is only used by Java developers

★ JRE → Environment to only run (not develop)
★ JVM → It is responsible for executing the Java program line by line ∴ also known as interpreter It is part of both JDK & JRE



JDK = JRE + Development tools
JRE = JVM + library

Functions

- Building blocks of java program
- main() is the first function to be executed

- Syntax:-

  return_type function_name (argument list)

- Prototyping required but variable name not necessary

Program:-

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

optional

class object function

- ☆ You should start class name with capital letter
- ☆ Predefined classname starts with capital letter.
- ☆ Save the file as main class name.

Command to Compile:- javac Hello.java
" " Execute:- java Hello

Program (Hello.java) → javac Hello.java → Hello.class

- ☆ The name of bytecode file/ class file will always be "classname.class" regardless of the name you give to program files
- ☆ Classfile is used at runtime.

⊙ If there are two classname in a program then after compilation two class files are created for both the classes (Ex :- abc.class , xyz.class)

Example

```
class abc
{
    public static void main(String[]args)
    {
        system.out.println("Hello !\n\n");
        System.out.println("Java!");
    }
}

class xyz
{
}
```

☆ when we command → java abc

o/p → Java
Hello

☆ But if we command → java xyz

we get an exception - "main thread not found" as xyz does not have main().

⊙ the byte code file that we are used to exec the program is the file that contains the main()

☆ 50 classes in program 50 byte codes

☆ Multiple classes can have main method without any error during compilation.

☆ All classes in the program are independent from each other

When we give command java abc, main of abc will be executed

☆ In single class you can't create more than one main method with same argument else it is compile time error

Example :- class xyz
{
public static void main (String[] args)
{

}
PSVM (String[] args)
{
}
}

This will show error as main have same arguments

☆ But main function can be overloaded in Java

Ex :- class xyz {
PSVM (String[] args)
{ ---
---- }
PSVM ()
{
}
}

* here no error as arguments in both main are different

* But only main() which has arguments will be executed at the run time

⑥ Datatypes

They specify the different sizes & values that can be stored in the variable

Two types :-

1. Primitive :- It includes boolean, char, byte, short, int, long, float & double

2. Non-Primitive :- Includes classes, Interfaces, Arrays.
They can be classified as value type and Reference type

@ Value type :- Variables which directly holds the value

@ Reference type :- Variable which holds the address of memory location where actual value is stored.

☆ Java doesn't support Call by reference

☆ Objects in java are "reference type" by default.

L-6

⑥ Example Programs

```
class Exam
{
    Public static void main.(string[] x)
    {
        int a;
        int b;
        a=12;
        b=14
        int c;
        c=a+b;
        System.out.println("sum"+c);
    }
}
```

⑦ Type Safety :- A language is type safe if most of the errors are caught by the compiler.

* If we add two bytes type variables (a+b) in java, it may increase its size therefore the result can't be stored in a byte type variable else it will show Compile time Error.

⇒ If we take float a=10.4, float b=9.6 & float c=a+b then also time is compile time error

as by default Java supports double Precision but float is of single precision

Solution :- whenever you declare float type variables then add F or f as postfix to the value

* No need to write d or D after double type value for double precision as it is used by default.

OPERATORS

All c/c++ operators are used in java

1. Arithmetic Operator
2. Relational "
3. Bitwise "
4. Unary "
5. Ternary "
6. Logical "

⑥ Control Statement

1. Sequential CS → Line by line all executed
2. Conditional CS → Only statement with condition (if else)
3. Iterative loop CS → multiple times
4. Switch case CS

→

```
Class Hello {
    PSVM (String [] x)
    {
        int a=5;
        if (a)
            System.out.println("Hello");
        else
            System.out.println("Bye");
    }
```

* The previous ex will give error as Java supports Boolean Datatypes to compute true and false.

⑥ Create Boolean Variable

```
boolean Javafun = true;
boolean ourday = false;

system.out.println (Javafun); //op true
system.out.println (ourday); //op false
```

Command line Arguments

The argument which is passed at runtime is known as Command line argument.

⇒

```
class cmdla
{
    PSVM (String [] args) → name of array (object)
    {                  ↳ array of strings
        int len i=0;
        len = args.length;
        for(i=0; i<len ; i++)
            System.out.println (args[i]);
    }
}
```

To compile :- javac cmdla.java

__/__/__

Enhanced For Loop

The syntax of the java for each loop is :-

```
for(datatype item : array) { }
```

→ It takes the value from the array/collection till the entire value is taken out
→ No termination condition
→ datatype should be same as array's

→
```
class Cmdla
{
    PSVM(String[] args)
    {
        for(String s : args)
            System.out.println(s);
    }
}
```

★ An object is a reference type

It holds the address of memory location where actual value is stored

Ques. WAP to accept 2 values & add them

```
class Abc
{  psvm (String[] k)
   {
      int a,b,c;
      a = k[0];
      b = k[1];
      c = a+b;
      System.out.println("Sum"+ c);
   }
}
```

Command :- Javac Abc.java
Java Abc 10,20

* exception will come - as value of a is stored in k[0] but a is int type & k is string type.

* even after typecasting it'll give error as "Strings" is a class not datatype & k is object.

To solve this problem → we use wrapper class

WRAPPER CLASS

It provides the mechanism to convert primitive to object & object to primitive

* means conversion of value type to reference type & vice versa

Since J2SE 5.0 unboxing and autoboxing feature do this conversion automatically

Primitive to object → Autoboxing
Object to primitive → Unboxing.

Primitive type	Wrapper class
boolean	Bodean
char	Character
byte	Byte
short	Short
int	Integer
long	long
float	Float
double	Double

◎ Example :-

```
Class Abc
{  psvm (String [] k)
   {
      int a, b, c;
      a = Integer.parseInt(k[0]);
      b = Integer.parseInt(k[1]);
      c = a+b;
      System.out.println("Sum=" +c);
   }
}
```

Command :- Javac Abc.Java
Java Abc 12 23
o/p = 35

◎ To see name of method

Command → javap java.lang.Integer → wrapper class
(Package (just like header file))
↓
All methods of class

L-8

★◎ Scanner Class ★ used to accept input values from user
It is available in 'util' package.

★ Lang package gets automatically imported in java, all other have to be imported seperately

To use scanner class, create an object of the class and use any of the available methods found in the scanner class documentation

__/__/____

◎ Example :-

```
import java.util.*;
class main {
        PSVM (String [] args)
        {
        Scanner myobj = new Scanner (System.in);
        System.out.println ("Enter Username");
        String username = myobj.nextLine();

        System.out.println ("username is: "+username)
        }
        }
```

* Methods of Scanner Class

◎	Method	description
1.	nextBoolean()	Reads a boolean value from the user
2.	nextByte()	Reads a byte value from the user
3.	nextDouble()	Reads a double value from user
4.	nextFloat()	" " Float " " "
5.	nextInt()	" " Int " " "
6.	nextLine()	" " String " " "
7.	nextLong()	" " Long " " "
8.	nextShort()	" " Short " " "

© Example :-

```
Export java.util.*;

class main
{
PSVM (String[] args)
{
Scanner myobj = new Scanner (System.in);
System.out.println ("Enter the name, age & salary");

String name = myobj.nextLine();
int age = myobj.nextInt();
double salary = myobj.nextDouble();

System.out.println("Name = " + name);
System.out.println ("Age = " + age);
System.out.println("Salary = " + salary);
}
}
```

L-9

⑤ Add method

Syntax :-

```
returntype methodname(Argumentlists)
{
    ------
    ------
}
```

Ex :-

```
class Abc
{
    void disp()
    { System.out.println("Hello");
      System.out.println("How are you?");
    }
    PSVM(String []args)
    {
      System.out.println("Welcome");
    }
}
```

o/p ⇒ Welcome

⑥ Call method

To call a method we need to make the object of class

Syntax :- `classname objname;`

* But if you call this object it will show null exception as no memory is allocated.

__/__/____

* To allocate memory we need to instantiation of object.

⇒ objectname = new classname();

or

classname obj = new classname();

⑥ Example

```
Class Abc
{
  Void disp()
  {
    System.out.println("Hello");
    System.out.println("How are you?");
  }
  PSVM (String[]x)
  {
    System.out.println("welcome");
    Abc obj1;
    obj1 = new Abc();
    obj1.disp();
    Abc obj2 = new Abc();
    obj2.disp();
    System.out.println("End");
  }
}
```

☆ Class object can be made inside any class.

Example :-

```
Import java.util.*;
class Abc
 {
   void add()
   {
   Scanner S1 = new Scanner(System.in);
   int a, b, c;
   System.out.println("Enter two Values");
    a = S1.nextInt();
    b = S1.nextInt();
    c = a+b;
   System.out.println("Result=" + c);
   }
  Void sub()
   {
   Scanner S2 = new Scanner(System.in);
    int a, b, c;
    System.out.println("Enter the Values");
     a = S1.nextInt();
     b = S2.nextInt();
     c = a-b;
    System.out.println("Result=" + c);
   }

 Psvm(String [] args)
  {
    System.out.println("Welcome");
     Abc obj = new Abc();
     obj.add();
     obj.sub();
      System.out.println("End");
  } }
```

* To define variables in class scope we use Instance variables. Then you don't need to create variables in every method of class [like Global variable]

⇒ Class Abc
{
int a,b,c;
};

* We can't access Instance variables in main method it'll show errors - "Trying to access non-static variables in static reference

L-10

__/__/__

◎ Static properties of Methods in Java

* Static variables are the variables which does not allocate memory for each individual objects separately. It allocates memory once & shared by all objects regardless no of objects.

Example

```
class Abc
{
 int a;
 static int b;
 void get(int p)
 {
   a=p;
   b++;
 }
 void disp()
   {
   System.out.println("a="+a);
   System.out.println("b="+b);
   }
PSVM(String[]k)
 {
 Abc a1 = new Abc();
    a1 = new Abc();
 Abc a2 = new Abc();
    a2 = new Abc();
 Abc a3 = new Abc();
    a3 = new Abc();
 a1.get(0);
 a1.disp();
```

```
a2.get(20);
a1.disp();
a2.disp();
a3.get(30);
a1.disp();
a2.disp();
a3.disp();
}
}
```

o/p = 10 1
20 2 2
30 3 3 3

☆ Static Method

* Static method are the method which can only access static members inside this
* Static method is called by using the class name instead of object.

Ex:-

```
class ABC{
int a;
Static int b;
void get(int p);
{
a=p;
b++;
}
void disp()
{
System.out.println("a="+a);
" " " ("b="+b);
}
```

```
static void show()
{
  // System.out.println("a= "+a);
     System.out.println("b= "+b);
}
  PSVM (String[]k)
  {
    Abc a1 = new Abc();
       a1= new Abc();
    Abc a2 = new Abc();
       a2 = new Abc();
    Abc a3 = new Abc();
       a3 = new Abc();
    a1.get(10);
    a1.disp();
    a2.get(10);
    a1.disp();
    a2.disp();
    a3.get(30);
    a3.disp();
    a2.disp();
    a3.disp();

    show(); // you can call directly as mass
              // is static.
    abc.show();
  }
}
```

⑥ Array in Java

① dataType[] arrayRefVar;
or
dataType arrayRefVar[];

arrayRefVar = new datatype[arraySize];

for example :-

Int []arr;
Arr = new int[5];
or
Int []arr = new int[5];
or
Int []arr = {1,5,7,9,11};