**OPP’s with JAVA INTERVIEW QUESTION – 1**

1. **What do you know about JVM, JDK, JRE ?**

Ans :- **JDK** is the kit for development of program

We have to download and install it on machine

It provide JRE (java runtime environment) and development tools

**Java Development Tools + Java Docs + rt.jar + JVM**

**JRE** is runtime environment which provide the set of libraries and files

**rt.jar** and **JVM** are integrated part of JRE

JRE is a software which comes with JDK

**Java Development Tools + Java Docs + JRE[ rt.jar + JVM ]**

**JVM** is an engine which manage execution of java applications

It provide runtime environment in which java bytecode can be executed

3 Components of JVM

1 Class Loader Sub System

2 Runtime Data Areas

3 Execution Engine

1. **Is JVM Platform dependent or independent?**

Ans :- JVM is Platform Dependent because the configuration of each OS is different from each other

1. **What is Bytecode?**

Ans :- Bytecode is the machine code in java

When compiler compiles the program it generate Bytecode which is in the form of .class file

1. **Write a “Hello Wold” program without giving semicolon?**

Ans :- **public** **class** Program {

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

System.***out***.println("Hello World")

}

}

1. **What is the meaning of System.out.println()**

Ans :- It is used to print an any statement

**System** is final class defined in the java.lang.package

**Out** is the static member field of the System class

**Println** is a public method of PrintStream Class

1. **What is method Overloading ?**

Ans :- Method Overloading means same method name but different parameters

If a class has multiple method having same name but different parameter

is known as method overloading

We can overload main method in JAVA

1. **Which are the rules of method overloading ?**

Ans :- Method name is same but different parameter

Different data types

Changing number of argument

Changing data type

1. **Can we overload main method ?**

Ans :- Yes we can overload main method, JVM invoke main method

But entry point is **public static void main ( String[] args )**

1. **What do you know about Wrapper Class ?**

Ans :- In java Primitive type are not classes but for every primitive type java has defined a class

it is called Wrapper Class

All Wrapper Class are final

Wrapper Class are declared in java.lang.package

1. **What is Boxing and Unboxing? Explain with Example**

Ans :- Boxing – It is a conversion of Primitive type to Non-Primitive Type

Ex. int num = 10;

String str = Integer.toString( number );

Unboxing – It is a conversion of Non-Primitive type to Primitive Type

Ex. String str = “10”;

int num = Integer.parseInt( str );

1. **Explain Class and Instance by giving real time example**

Ans :- Class – It is Template or BluePrint for object to describe behavior/state of that object

Ex.

Public class Dog {

String breed; int age; String color;

Void barking ( ) {

}

Void running ( ) {

}

}

Instance – It is variable to create instance of the class

It create object of class with new keyword

Ex.

Class Instance {

}

Class Program {

public static void main ( String[ ] args ) {

Instance i = new Instance( ); // Instantiation

}

}

1. **What is different Between Primitive and Non-Primitive Type**

Ans :- Primitive –

It is a Value Type

It Specify the size and the type of variable

It has no method

8 type of Primitive Data type

( Boolean, byte, short, char, int ,long, float,double )

Non- Primitive -

It is a Reference Type

It has null value

It start with uppercase letter

It is use for call the method to perform certain operations

1. **“final Complex c= new Complex( );” what is meaning of this statement**

Ans :- It means that reference of that instance is final

First instance is final instance of that reference

It means we can not give another new instance of that reference

Ex.

public class Program {

public static void main( String[ ] args ) {

final Complex c = new Complex(10, 20 );

c.setReal(11);

c.SetImag(22);

c = new Complex(20,10); // NOT OK

}

}

1. **What is Constructor Chaining ?**

Ans :- Constructor chaining is the process of calling one constructor from another constructor with respect to current object

Ex.

class Add {

void Add( ) {

this(10,20); //invokes parameterized constructor

}

Void add(int a, int b) {

System.out.println(a+b);

}

}

1. **What do you know about static initializer block ?**

Ans :- In static initializer block we can initialize the static field

Static initialization block can be used for static initializations of a class

Static block is executed only once

Ex.

class Test {

     static int i;

     int j;

     // start of static block

     static {

         i = 10;

         System.out.println("static block called ");

     }

}

class Main {

     public static void main(String args[]) {

         System.out.println(Test.i);

     }

}

Output:

Static block called

10

1. **Why static method do not get this reference ?**

Ans :- Static method can invoke without creating an object

Static method get loaded into memory

“this” keyword is used as a reference to an instance

Static method doesn’t have any instance that’s why we cannot use “this” reference within a static method

If we try to do then will get compiler error

public class Sample{

   static int num = 50;

   public static void demo(){

      System.out.println("Contents of the static method");

   }

   public static void main(String args[]){

      Sample.demo();

   }

}

Output:

Contents of the static method

public class Sample{

   static int num = 50;

   public static void demo(){

      System.out.println("Contents of the static method"+this.num);

   }

   public static void main(String args[]){

      Sample.demo();

   }

}

Output:

Compiler time error

1. **What is singleton class ? Can you write code for it ?**

Ans : - In object-oriented programming, a singleton class is a class that can have only one object (an instance of the class) at a time

After first time, if we try to instantiate the Singleton class, the new variable also points to the first instance created

So whatever modification we do in any variable inside the class through any instance it will affect the variable of the singleton instance created

Ex.

class Singleton{

private Singleton() {

}

private static Singleton instance = null;

public static Singleton getInstance( ) {

if( instance == null )

instance = new Singleton();

return instance;

}

}

public class Program {

public static void main(String[] args) {

Singleton s1 = Singleton.getInstance();

Singleton s2 = Singleton.getInstance();

Singleton s3 = Singleton.getInstance();

}

}

1. **Which are components of JVM ? Which are memory areas of JVM ?**

Ans:- JVM have 3 components which are followings

1. Class loader sub system
2. Runtime Data Areas
3. Execution Engine

Memory areas of JVM

1. Method area
2. Heap
3. Java stacks
4. pc registers
5. native method stacks
6. **What is Different between default and protected ?**

Ans :-

1. **What is class and class path ?**

Ans :-

1. **What do you know about package ?**

Ans :-

1. **Why we can not multiple public classes in single.java file**

Ans :- There **can** be only **one public class** in a **java file** because the name of **java file is** same as the name of **public class**. And obviously **we can**'t have a **file** with **two** different names

1. **What is the different between import and static import**

Ans :-

1. **Which are the methods of java.lang.object class**

Ans :-