# * JAVA Interview Questions
## Assignment - 1

Q.1) What do you know about JVM, JRE and JDK ?

Ans: Full form of all 3 are:

1) JDK : Java Development Kit

2) JRE : Java Runtime Environment

3) JVM : Java Virtual Machine

→ Definition & and Concept

- JDK is a software development environment used for developing Java apps. It includes JRE, an interpreter (Java), a compiler (javac), an archiever (jar), a document generator (Javadoc) and other tools.

- JDK is basically a kit that provides an environment to develop & execute Java program.

JDK = JRE + dev tools
(to execute program) (provide environment to develop program)

→ JRE is also known as Java RTE (Java RunTime Environment). It provides the minimum requirements for executing a Java application. It consists of JVM, core classes and other supporting files.

JRE = JVM + set of + other libraries supporting files

– JRE is a package that provides environment to only run the program and not execute it.

→ JVM is called a virtual machine because it physically doesn't exist. It is an important part of both JDK and JRE because its inbuilt in both. The java program run by user goes into JVM and JVM executes the java program line by line, i.e works as an interpreter.

→ JVM performs following tasks:

1.) Load the code
⇓
2.) Verifies the code
⇓
3.) Executes the code
⇓
4.) Provides runtime environment

Q.2) Is JRE platform dependent or independent ?

Ans: JRE = JVM + set of + other support libraries files

→ JRE contains and other files to run

– JRE contains files required to run Java programs in JVM. It contains platform dependent files which are used to create JVM in the operating system so JRE is platform dependent.

Q.3 Which is ultimate base class in java class hierarchy ? List the name of methods of it.

Ans: Ultimate base class is also known as topmost class. In Java, Object class is the parent class of all classes by default.

– java.lang.Object
⇓ ⇓
Package Class

→ Some imp methods of this class are:

1.) equals(): It compares two objects & does the shallow comparision and returns value in boolean form (true or false).

Sign: `boolean equals(Object obj)`

2.) hashCode(): Its a hashkey associated with each object. Hashcode is not address of object and it can be negative.

Sign: `public int hashCode()`

3.) toString(): returns ClassName@HashCode in hexadecimal form.

Sign: `public String toString()`

4.) finalize(): is invoked by the garbage collector before object is being garbage collected.

Sign: `public void finalize()`

Q.4 What are reference types in Java?

Ans: In Java, non-primitive data types are known as reference types. Reference data types refers to where data is stored.

- Reference data types contain the address or reference of dynamically created objects.

Eg: If Demo is a class and we create its object Dobj, then here Dobj is called reference type.

- All reference types are a subclass of type java.lang.Object.

→ Types of references types in Java:

1.) Array
2.) Class
3.) String
4.) Interface
5.) Enum (Enumeration)
6.) Annotation

1. **Array:** Array in Java is an object which contains elements of similar data type. The elements of an array are stored in continuous memory location.

→ dynamically create array:

```
int arr[] = new int[size];
```

→ **Advantages:**

1. It makes code optimized by retrieving & sorting the data efficiently.
2. We can randomly access any data located at an index position.

→ **Disadvantages:**

1. We can store only the fixed size of elements in array.
2. Its size doesn't grow at runtime.

→ **Real-life examples:**

– If we want to ~~create~~ store names of 100 employees of a company then we have to create an array of type string that can store 100 names.

```
String[] arr = new String[100];
```

2. **Class:** A class is a group of objects which have common properties. Its a blueprint from which objects are create. Its a logical entity.

class = data members + methods

eg: class <class_name> {

```
    data member;
    method;
}
```

→ **Advantages:**

1. Encapsulation
2. Code reusability
3. Inheritance
4. Polymorphism
5. Modularity

→ **Real-life examples:**

– Consider car as a class that has characteristics like name, colour, wheels etc. and behaviour like driving, braking etc.

– Also we can take laptop as class and Asus, Lenovo as objects of class laptop.

classmate
Date
Page

3. **Interface**: Interface in Java is a blueprint of a class. All data inside interface is public, final and static and all methods inside interface are public and abstract.

- 'interface' is a keyword used to define any interface.
- All methods in interface are abstract.
- It is used to achieve abstraction and multiple inheritance in Java.

eg:
```
interface <interface-name> {
    // data mem (final & static)
    // methods (abstract)
}
```

- **Advantages**:
1) Easier to test
2) Increased flexibility
3) Loosely coupled code
4) Faster development

Q.5 Explain widening & narrowing.

1) Widening (upcasting, implicit)

- assign smaller data type to larger data type

- larger data type = smaller data type
  long = int (assg int to long)

2) Narrowing (downcasting, explicit)

- assign larger data type to smaller data type

- smaller d.t = large d.t
  int = long
  (assign long to int)

Q.5 * Widening

```
int i = 9;
double d = i;   // d = 9.0
```

* Narrowing

```
double d = 9.81d;
int i = (int) d   // i = 9
                  (data loss)
```

Q.6 How will you print "Hello CDAC" statement on screen without semicolon?

Ans: Using 2 ways:

i) using if- else statements

```
class Demo1 {
    public static void main (String args[]) {
        if(System.out.printf("Hello CDAC")==null){
        }
    }
}
```

2.) Using equals method of String class

```
import java.util.*;

class Demo2 {
    public static void main (String args[]){

    if (System.out.append("Hello COAC").equals(nul
        {
            }
    }
}
```

Q.7 Can you write java application without main function? If yes, how?

Ans: Before Java 7, it was possible to run Java program without main function. User could easily write code in static block and code would run.

But after Java 7 or JDK 1.7, main() is mandatory. The compiler will first verify if main() is present or not. If program doesn't contain main method then we get an error — main method not found in class.

Q.9 In System.out.println, explain meaning of every word.

Ans: System: Its a final class defined in java.lang package.

out: Its an instance of PrintStream class which is public and static member of the System class.

println(): It prints any argument passed to it and adds a new line to it. It is a method of PrintStream class.

Q.10 How will you pass object to function by reference?

Ans: When an object is passed to function by reference, the reference itself is passed by use of call by value. Value being passed refers to the object then the copy of that value will still refer to the same object. So in Java, actually there is pass by value only.

Q.11 Explain constructor chaining. How can we achieve it in C++?

Ans: To call one constructor from another constructor body is called constructor chaining.

- It is done using this keyword, in Java.
- While calling constructor using this, it must be the very first statement inside the constructor.
- Constructor chaining in C++ is known as constructor delegation. Its allowed from C++ 11.

eg:
```
class Student {
    int roll;
    double marks;

    Student() {
    }

    Student(int roll) {
        this.roll = roll;
    }

    Student(int roll, double marks) {
        this(roll); // (will call one-argu construc. of student)
    }
```

→ this : 'this' is a keyword which keeps reference of current object.

classmate
Date
Page

Q.12 What are the rules to overload method in sub class?

Ans:

1.) Methods must have same name and different signatures.

2.) Return type may or may not be same but parameters must be different.

Method overloading is possible in same class as well as child class.

eg:
```
class First {
    void myFun(){
        //.... code
    }

    void myFun(int x) {  // (Overloading in same class)
        //.... code
    }
}

class Second extends First {
    void myFun(int x, int y) {
        //.... code
    }                          // (Overloading in child class)
}
```

Q.19 Explain upcasting.

Ans: Upcasting is the typecasting of a child object to a parent object.

Parent ref type = child ref. type

~~First f; // p~~

Parent p = new Child();

~~Child c = (Child) p; //~~

- Upcasting is done implicitly.

Q.21 What do you mean by final method?

Ans: 'final' is a keyword in Java used with data, method and class.

- If final is used with method that method cannot be overriden in child class.

eg:
```
class First {
    final void m1() {
        System.out.println("Final method");
    }
}

class Second extends First {
    void m1() {
        System.out.println("No final");
    }
}
```
Ln 10

o/p: Ln10 will give compile error. Cannot override.