Most Popular Java Interview Questions

Q #1) What is JAVA?
Ans: Java is a high-level programming language and is platform independent.
Java is a collection of objects. It was developed by Sun Microsystems. There are a lot of applications, websites and Games that are developed using Java.

Q #2) What are the features in JAVA?
Ans: Features of Java:

- **Oops concepts**
  - Object-oriented
  - Inheritance
  - Encapsulation
  - Polymorphism
  - Abstraction
- **Platform independent:** A single program works on different platforms without any modification.
- **High Performance:** JIT (Just In Time compiler) enables high performance in Java. JIT converts the bytecode into machine language and then JVM starts the execution.
- **Multi-threaded:** A flow of execution is known as a Thread. JVM creates a thread which is called main thread. The user can create multiple threads by extending the thread class or by implementing Runnable interface.

Q #3) How does Java enable high performance?
Ans: Java uses Just In Time compiler to enable high performance. JIT is used to convert the instructions into bytecodes.

Q #4) What are the Java IDE's?
Ans: Eclipse and NetBeans are the IDE's of JAVA.

Q #5) What do you mean by Constructor?
Ans: The points given below explain what a Constructor is in detail:

- When a new object is created in a program a constructor gets invoked corresponding to the class.
- The constructor is a method which has the same name as class name.
- If a user doesn't create a constructor implicitly a default constructor will be created.
- The constructor can be overloaded.
- If the user created a constructor with a parameter then he should create another constructor explicitly without a parameter.

Q #6) What is meant by Local variable and Instance variable?
Ans: Local variables are defined in the method and scope of the variables that have existed inside the method itself.
An instance variable is defined inside the class and outside the method and scope of the variables exist throughout the class.

Q #7) What is a Class?
Ans: All Java codes are defined in a class. A Class has variables and methods.
Variables are attributes which define the state of a class.
Methods are the place where the exact business logic has to be done. It contains a set of statements (or) instructions to satisfy the particular requirement.

Example:

```
public class Addition{ //Class name declaration
int a = 5; //Variable declaration
int b= 5;
public void add(){ //Method declaration
int c = a+b;
}
}
```

Q #8) What is an Object?

Ans: An instance of a class is called object. The object has state and behavior. Whenever the JVM reads the “new()” keyword then it will create an instance of that class.

Example:

```
public class Addition{
public static void main(String[] args){
Addion add = new Addition();//Object creation
}
}
```

The above code creates the object for the Addition class.

Q #9)What are the Oops concepts?

Ans: Oops concepts include:

- Inheritance
- Encapsulation
- Polymorphism
- Abstraction
- Interface

Q #10) What is Inheritance?

Ans: Inheritance means one class can **extend** to another class. So that the codes can be reused from one class to another class.
Existing class is known as Super class whereas the derived class is known as a sub class.

Example:

```
Super class:
public class Manupulation(){
}
Sub class:
public class Addition extends Manipulation(){
}
```

Inheritance is applicable for public and protected members only. Private members can’t be inherited.

Q #11) What is Encapsulation?

Ans: Purpose of Encapsulation:

- Protects the code from others.
- Code maintainability.

Example:

We are declaring ‘a' as an integer variable and it should not be negative.

```
public class Addition(){
int a=5;
}
```

If someone changes the exact variable as “***a = -5”*** then it is bad.

In order to overcome the problem we need to follow the below steps:

- We can make the variable as private or protected one.
- Use public accessor methods such as set<property> and get<property>.

So that the above code can be modified as:

```
public class Addition(){
private int a = 5; //Here the variable is marked as private
}
```

Below code shows the getter and setter.

Conditions can be provided while setting the variable.

```
get A(){

}

set A(int a){

if(a>0){// Here condition is applied

.........

}

}
```

For encapsulation, we need to make all the instance variables as private and create setter and getter for those variables. Which in turn will force others to call the setters rather than access the data directly.

Q #12) What is Polymorphism?

Ans: Polymorphism means many forms.

A single object can refer the super class or sub-class depending on the reference type which is called polymorphism.

Example:

```
Public class Manipulation(){ //Super class
public void add(){
}
}
public class Addition extends Manipulation(){ // Sub class
public void add(){
}
public static void main(String args[]){
Manipulation addition = new Addition();//Manipulation is reference type and Addition is
addition.add();
}
}
```

Using Manipulation reference type we can call the Addition class “add()” method. This ability is known as Polymorphism.

Polymorphism is applicable for **overriding** and not for **overloading**.

Q #13) What is meant by Method Overriding?

Ans: Method overriding happens if the sub class method satisfies the below conditions with the Super class method:

- Method name should be same

- Argument should be same
- Return type also should be same

The key benefit of overriding is that the Sub class can provide some specific information about that sub class type than the super class.

Example:

```
public class Manipulation{ //Super class

public void add(){

………………

}

}



Public class Addition extends Manipulation(){

Public void add(){

………..

}

Public static void main(String args[]){

Manipulation addition = new Addition(); //Polimorphism is applied

addition.add(); // It calls the Sub class add() method

}

}
```

addition.add() method calls the add() method in the Sub class and not the parent class. So it overrides the Super class method and is known as Method Overriding.

Q #14) What is meant by Overloading?

Ans: Method overloading happens for different classes or within the same class.

For method overloading, subclass method should satisfy the below conditions with the Super class method (or) methods in the same class itself:

- Same method name
- Different argument type
- May have different return types

Example:

```
public class Manipulation{ //Super class

public void add(String name){ //String parameter

………………

}

}



Public class Addition extends Manipulation(){

Public void add(){//No Parameter

………..

}

Public void add(int a){ //integer parameter



}

Public static void main(String args[]){

Addition addition = new Addition();

addition.add();

}

}
```

Here the add() method having different parameters in the Addition class is overloaded in the same class as well as with the super class.

Note: Polymorphism is not applicable for method overloading.

Q #15) What is meant by Interface?

Ans: Multiple inheritance cannot be achieved in java. To overcome this problem Interface concept is introduced.

An interface is a template which has only method declarations and not the method implementation.

Example:

```
Public abstract interface IManupulation{ //Interface declaration
Public abstract void add();//method declaration
public abstract void subtract();
}
```

- All the methods in the interface are internally **public abstract void**.
- All the variables in the interface are internally **public static final** that is constants.
- Classes can implement the interface and not extends.
- The class which implements the interface should provide an implementation for all the methods declared in the interface.

```
public class Manupulation implements IManupulation{ //Manupulation class uses the interfa
Public void add(){
……………
}
Public void subtract(){
…………….
}
}
```

Q #16) What is meant by Abstract class?

Ans: We can create the Abstract class by using “Abstract” keyword before the class name. An abstract class can have both “Abstract” methods and “Non-abstract” methods that are a concrete class.

Abstract method:

The method which has only the declaration and not the implementation is called the abstract method and it has the keyword called “abstract”. Declarations are the ends with a semicolon.

Example:

```
public abstract class Manupulation{
public abstract void add();//Abstract method declaration
Public void subtract(){
}
}
```

- An abstract class may have a Non- abstract method also.
- The concrete Subclass which extends the Abstract class should provide the implementation for abstract methods.

Q #17) Difference between Array and Array List.

Ans: The Difference between Array and Array List can be understood from the below table:

Array	Array List
Size should be given at the time of array declaration. String[] name = new String[2]	Size may not be required. It changes the size dynamically. ArrayList name = new ArrayList
To put an object into array we need to specify the index. name[1] = “book”	No index required. name.add(“book”)

Array	Array List
Array is not type parameterized	ArrayList in java 5.0 are parameterized. Eg: This angle bracket is a type parameter whi means a list of String.