Java OOPS Interview Questions and Answers

Java OOPS Interview Questions and Answers 36 Votes

Hits: 16706

1. **What are JAVA Byte Codes?**

   Java bytecode is an intermediate language which is typically compiled from Java.

2. **What is JVM (Java Virtual Machine) ?**

   JVM stants for Java Virtual Machine which is a run time environment for the compiled java class files.JVM can execute only JAVA bytecode.

3. **What is JIT (Just-in-Time) Compilation ?**

   There are two ways a language can be compiled

   a) ***Compiled Language***
   b) ***Interpreted Language***

   A machine understats only banary language, so fanally a source code has to be compiled in binary format. In compiled way the compiler directly generate the binary file from source code.While in interpreted way it generate the class file which is then run by virtual machine.That means binary file is generated at the run time (compilation is done on need basis) this type of compilation is called JIT (Just-in-Time) compilation.

4. **What is Object Oriented Programming?**

   Object-oriented programming (OOP) is a programming paradigm or a problem solving technique.
   Object-oriented programming maps the programming model to real world concept.This is technique to think real world in terms of object.
   An object-oriented program may be viewed as a collection of interacting objects, as opposed to the conventional model, in which a program is seen as a list of tasks (subroutines) to perform.
   In OOP, each object is capable of receiving messages, processing data, and sending

messages to other objects.

5. **What’s a Class?**

   A Class describes all the attributes of object, as well as the methods that implements the behavior of member object. That means this is a template or blue print.
   In object-oriented programming , a class is a template definition of the method s and variable s in a particular kind of object . Thus, an object is a specific instance of a class; it contains real values instead of variables.

   ```
   public class Person {
       String name;
       String age;

       public String getAge() {
           return age;
       }

       public void setAge(String age) {
           this.age = age;
       }

       public String getName() {
           return name;
       }

       public void setName(String name) {
           this.name = name;
       }
   }
   ```

6. **What’s an Object?**

   Object is an instance of a class, it contains real values instead of variables.

   ```
   Person employee=new Person();
   ```

7. **What’s the relation between Classes and Objects?**

   Class is a defenition , where as object is an instance of a class created.Class is a blue print while Objects are actual objects existing in the real world. Eg: Suppose person is a class- John, Sara.. are objects of class person.

8. **What's singleton class ?**

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

9. **What's method In object-oriented programming ?**

   In object-oriented programming, a method is a programmed procedure that is defined as part of a class and included in any object of that class.

10. **What are the funtamental principles of Object-oriented systems?**

    Abstraction, Encapsulation, Inheritance, Association, Aggregation, Polymorphism

11. **What is Abstraction?**

    Abstraction is separating the functions and properties that logically can be separated to a separate entity.
    Abstraction in Object Oriented Programming helps to hide the irrelevant details of an object.

12. **Define Encapsulation**

    Process of hiding all internal details of an object from out side the world.Encapsulation is a language mechanism to restrict the access of the Objects components to other Objects or Classes.

13. **What is Inheritance ?**

    In object-oriented programming (OOP), inheritance is a way to reuse code of existing objects.
    A class can inherit attributes and behavior from pre-existing class called base classes, superclasses, or parent classes. The resulting class is known as derived classes, subclasses, or child classes.

14. **What is Polymorphism ?**

    The occurrence of something in different forms. In object-oriented programming, polymorphism refers to a programming language's ability to process objects differently depending on their data type or class. More specifically, it is the ability to redefine methods for derived classes.

15. **What is Overloading ?**

    Overloading is one type of polymorphism. Overloading allows an object to have different meanings depending on its context. The term is used most often in reference to operators that can behave differently depending on the data type, or class, of the operands.

16. **How do you implement inheritance in Java?**

    In Java, Inheritance is Implemented by "EXTEND" keyword.

```
public class employee extends person {

}
```

17. **How can we implement polymorphism in Java?**

    Java has excellent support of polymorphism in terms of
    a) **Method Polimorphism** through **overloading** and **overriding**.
    b) **Object polymorphis**m through **inheritance** / **Interface**

18. **Define overriding in Java**

    when you extend a class and write a method in the derived class which is exactly similar to the one present in the base class, it is termed as overriding.

```
public class Vehicle {
    public String getEngineType(){
        //some code here
    }
}
public class Vehicle {
public String getEngineType(){
        //some new code here
    }
}
```

    As you can see, in the class Vehicle , we have overridden the method present in the BaseClass (Vehicle) with a completely new piece of code.

19. **Define overloading in Java**

    when you have more than one method with the same name but different arguments, the methods are said to be overloaded.

```
Public void saveUserInfo(int userId){

}
public void saveUserInfo(String userName,String Location){

}
```

20. **Differenciate method overriding and overloading in Java ?**

Method overloading deals with the notion of having two or more methods(functions) in the same class with the same name but different arguments.

While Method overriding means having two methods with the same arguments, but different implementation. One of them would exist in the Parent class (Base Class) while another will be in the derived class(Child Class).@Override annotation is required for this.

21. **What's an interface ?** In the Java programming language, an interface is a reference type, similar to a class, that can contain only constants, method signatures, and nested types. There are no method bodies. Interfaces cannot be instantiated—they can only be implemented by classes or extended by other interfaces.

```
public interface Person {
    // constant declarations, if any
    String name;
    public String getName();
    public void setName(String iName);
}
```

22. **How can we implement an interface in java ?**

    To use an interface, you write a class that implements the interface. When class implements an interface, it provides a method body for each of the methods declared in the interface

23. **What is an Abstract class ?**

    Abstract classes in Java are classes which cannot be instantiated, meaning you cannot create new instances of an abstract class.
    The purpose of an abstract class is to function as a base for subclasses.
    You can declare that a class is abstract by adding the **abstract** keyword to the class declaration.

    Here is an example:

```
public abstract class MyAbstractClass {

}
```

24. **What are Abstract methods ?**
    An abstract class can have abstract methods. You declare a method abstract by adding the **abstract** keyword in front of the method declaration.
    Adbtract method do not have implementation.
    Abstract method should be implemented in the sub class which inherit them.

```
public abstract class MyAbstractClass {
    public abstract void abstractMethod();
}
```

25. **What’s the difference between 'Abstract' classes and 'Interfaces' ?**
1) Absract Class can only be inherited, while interface cannot be. Interface has to be implemented.
2) Abstract class can have implementation ,where as interface cannot implement methods.

26. **What’s difference between Static and Non-Static fields of a class ?**

Non static values are also called instance variables. Each object of class will be having its own copy of non static instance variable. Static variable will be having only one copy of instance variable which is shared among the objects of the class.

```
public class userLogin {
    // constant declarations, if any
    public int userid; // Static variable declaraton
    public static int userCount; // Non Static variable declaration
}
```

27. **What are inner classes ?**
The Java programming language allows you to define a class within another class. Such a class is called a nested class and is illustrated here:

```
class OuterClass {
    ...
    class NestedClass {
        ...
    }
}
```

Nested classes are divided into two categories: static and non-static. Nested classes that are declared static are simply called static nested classes. Non-static nested classes are called inner classes.

28. **Why we are using inner classes ?**

There are several compelling reasons for using nested classes:

- It is a way of logically grouping classes that are only used in one place.
- It increases encapsulation.

- Nested classes can lead to more readable and maintainable code.

29. **What are packages ?**

    Package groups related classes and interface together and thus avoiding any name conflicts. Classes are grouped together in a package using the keyword 'package'

    ```
    package administration;
    public class userInfo {

    }
    public class departmentInfo {

    }
    ```

30. **What is a constructor in class ?**
    A constructor is a special method that is used to initialize a newly created object.
    The name of the constructor will be same as the class name. Constructor don't have a return type.

    ```
    public class userInfo {
        int userCount;
        userInfo(){
            userCount=0;
        }
        userInfo(int count){
            userCount=count;
        }
    }
    ```