Object Oriented programming Interview Questions

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

Object-Oriented Programming(OOPs) is a type of programming that is based on objects rather than just functions and procedures. Individual objects are grouped into classes. OOPs implements real-world entities like inheritance, polymorphism, hiding, etc into programming. It also allows binding data and code together.

**2. Why use OOPs?**

* OOPs allows clarity in programming thereby allowing simplicity in solving complex problems
* Code can be reused through inheritance thereby reducing redundancy
* Data and code are bound together by encapsulation
* OOPs allows data hiding, therefore, private data is kept confidential
* Problems can be divided into different parts making it simple to solve
* The concept of polymorphism gives flexibility to the program by allowing the entities to have multiple forms

**3. What are the main features of OOPs?**

* Inheritance
* Encapsulation
* Polymorphism
* Abstraction

**4) What is Inheritance?**

**Answer:** Inheritance means one class can extend to another class. So that the codes can be reused from one class to another class. The existing class is known as the 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 only applicable to the public and protected members only. Private members can’t be inherited.

**5) What is Encapsulation?**

**Answer: 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

**6) What is Polymorphism?**

**Answer:** Polymorphism means many forms.

A single object can refer to 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 reference type  addition.add();  }  } |

Using the 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**.

**7) What is meant by Method Overriding?**

**Answer: Method overriding happens if the sub-class method satisfies the below conditions with the Super-class method:**

* Method name should be the same
* The argument should be the same
* Return type should also be the 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.

**8) What is meant by Overloading?**

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

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

* Same method name
* Different argument types
* There may be 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 has different parameters in the Addition class is overloaded in the same class as with the super-class.

**Note:** Polymorphism is not applicable for method overloading.

**9) What is meant by Interface?**

**Answer:** Multiple inheritances cannot be achieved in java. To overcome this problem the 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 interface  Public void add(){  ……………  }  Public void subtract(){  …………….  }  } |

**10) What is meant by Abstract class?**

**Answer:** We can create the Abstract class by using the “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 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.

11**) Difference between overriding and overloading in java?**

Overriding Overloading In overriding method names must be same In overloading method names must be same Argument List must be same Argument list must be different atleast order of arguments. Return type can be same or we can return covariant type. From 1.5 covariant types are allowed Return type can be different in overloading. We cant increase the level of checked exceptions. No restrictions for unchecked exceptions In overloading different exceptions can be thrown. A method can only be overridden in subclass A method can be overloaded in same class or subclass Private,static and final variables cannot be overridden. Private , static and final variables can be overloaded. In overriding which method is called is decided at runtime based on the type of object referenced at run time In overloading which method to call is decided at compile time based on reference type. Overriding is also known as Runtime polymorphism, dynamic polymorphism or late binding Overloading is also known as Compile time polymorphism, static polymorphism or early binding.

12) **What is ‘IS-A ‘ relationship in java?**

‘is a’ relationship is also known as inheritance. We can implement ‘is a’ relationship or inheritance in java using extends keyword. The advantage or inheritance or is a relationship is reusability of code instead of duplicating the code. Ex : Motor cycle is a vehicle Car is a vehicle Both car and motorcycle extends vehicle.

13) **What is ‘HAS A’’ relationship in java?**

‘Has a ‘ relationship is also known as “composition or Aggregation”. As in inheritance we have ‘extends’ keyword we don’t have any keyword to implement ‘Has a’ relationship in java. The main advantage of ‘Has-A‘ relationship in java code reusability.

14) **Difference between ‘IS-A’ and ‘HAS-A’ relationship in java?**

IS-A relationship HAS- A RELATIONSHIP Is a relationship also known as inheritance Has a relationship also known as composition or aggregation. For IS-A relationship we uses extends keyword For Has a relationship we use new keyword Ex : Car is a vehicle. Ex : Car has an engine. We cannot say Car is an engine The main advantage of inheritance is reusability of code The main advantage of has a relationship is reusability of code.

15 ) **What are packages in java?**

Package is a mechanism to group related classes ,interfaces and enums in to a single module. Package can be declared using the following statement : Syntax : package Coding Convention : package name should be declared in small letters. package statement defines the namespace. The main use of package is 1) To resolve naming conflicts 2) For visibility control : We can define classes and interfaces that are not accessible outside the class.

**16) Can we have more than one package statement in source file ?**

We can’t have more than one package statement in source file. In any java program there can be atmost only 1 package statement. We will get compilation error if we have more than one package statement in source file. 41) Can we define package statement after import statement in java? We can’t define package statement after import statement in java. package statement must be the first statement in source file. We can have comments before the package statement.

17) **What are identifiers in java?**

Identifiers are names in java program. Identifiers can be class name, method name or variable name. Rules for defining identifiers in java: 1) Identifiers must start with letter,Underscore or dollar($) sign. 2) Identifiers can’t start with numbers . 3) There is no limit on number of characters in identifier but not recommended to have more than 15 characters 4) Java identifiers are case sensitive. 5) First letter can be alphabet ,or underscore and dollar sign. From second letter we can have numbers . 6) We should’nt use reserve words for identifiers in java.

18) **What are access modifiers in java?**

The important feature of encapsulation is access control. By preventing access control we can misuse of class, methods and members. 16 A class, method or variable can be accessed is determined by the access modifier. There are three types of access modifiers in java. public,private,protected. If no access modifier is specified then it has a default access.

19) **What is the difference between access specifiers and access modifiers in java?**

In java we have access specifiers as public,private,protected and default and access modifiers as static, final. But there is no such divison of access specifiers and access modifiers in java. In Java we have access modifiers and non access modifiers. Access Modifiers : public, private, protected, default Non Access Modifiers : abstract, final,.

20) **What access modifiers can be used for class ?**

We can use only two access modifiers for class public and default. public: A class with public modifier can be visible 1) In the same class 2) In the same package subclass 3) In the same package nonsubclass 4) In the different package subclass 5) In the different package non subclass. default : A class with default modifier can be accesed 1) In the same class 2) In the same package subclass 3) In the same package nonsubclass 4) In the different package subclass 5) In the different package non subclass.

21) **Explain what access modifiers can be used for methods?**

We can use all access modifiers public, private,protected and default for methods. public : When a method is declared as public it can be accessed 6) In the same class 7) In the same package subclass 8) In the same package nonsubclass 9) In the different package subclass 10) In the different package non subclass. default : When a method is declared as default, we can access that method in 1) In the same class 2) In the same package subclass 3) In the same package non subclass We cannot access default access method in 1) Different package subclass 2) Different package non subclass. protected : When a method is declared as protected it can be accessed 1) With in the same class 2) With in the same package subclass 3) With in the same package non subclass 4) With in different package subclass It cannot be accessed non subclass in different package. private : When a method is declared as private it can be accessed only in that class. It cannot be accessed in 1) Same package subclass 2) Same package non subclass 3) Different package subclass 4) Different package non subclass.

22) **Explain what access modifiers can be used for variables?**

We can use all access modifiers public, private,protected and default for variables. public : When a variables is declared as public it can be accessed 1) In the same class 2) In the same package subclass 3) In the same package nonsubclass 4) In the different package subclass 5) In the different package non subclass. default : When a variables is declared as default, we can access that method in 1) In the same class 2) In the same package subclass 3) In the same package non subclass We cannot access default access variables in 4) Different package subclass 5) Different package non subclass. protected : When a variables is declared as protected it can be accessed 1) With in the same class 2) With in the same package subclass 3) With in the same package non subclass 4) With in different package subclass It cannot be accessed non subclass in different package. private : When a variables is declared as private it can be accessed only in that class. It cannot be accessed in 1) Same package subclass 2) Same package non subclass 3) Different package subclass 4) Different package non subclass.

23) **What is final access modifier in java?**

final access modifier can be used for class, method and variables. The main advantage of final access modifier is security no one can modify our classes, variables and methods. The main disadvantage of final access modifier is we cannot implement oops concepts in java. Ex : Inheritance, polymorphism. final class : A final class cannot be extended or subclassed. We ar e preventing inheritance by marking a class as final. But we can still access the methods of this class by composition. Ex: String class final methods: Method overriding is one of the important features in java. But there are situations where we may not want to use this feature. Then we declared method as final which will print overriding. To allow a method from being overridden we use final access modifier for methods. final variables : If a variable is declared as final ,it behaves like a constant . We cannot modify the value of final variable. Any attempt to modify the final variable results in compilation error. The error is as follows “final variable cannot be assigned.”

24) **Explain about abstract classes in java?**

Sometimes we may come across a situation where we cannot provide implementation to all the methods in a class. We want to leave the implementation to a class that extends it. In such case we declare a class as abstract.To make a class abstract we use key word abstract. Any class that contains one or more abstract methods is declared as abstract. If we don’t declare class as abstract which contains abstract methods we get compile time error. We get the following error. “The type must be an abstract class to define abstract methods.” Signature ; abstract class { } For example if we take a vehicle class we cannot provide implementation to it because there may be two wheelers , four wheelers etc. At that moment we make vehicle class abstract. All the common features of vehicles are declared as abstract methods in vehicle class. Any class which extends vehicle will provide its method implementation. It’s the responsibility of subclass to provide implementation. The important features of abstract classes are : 1) Abstract classes cannot be instantiated. 2) An abstract classes contains abstract methods, concrete methods or both. 3) Any class which extends abstract class must override all methods of abstract class. 4) An abstract class can contain either 0 or more abstract methods.