1. What is the main difference between a class and an object?

The difference between class and an object are: -

|  |  |
| --- | --- |
| **Class** | **Object** |
| A class is a blueprint from which you can create the instance, i.e., objects. | An object is the instance of the class, which helps programmers to use variables and methods from inside the class. |
| A class is used to bind data as well as methods together as a single unit. | Object acts like a variable of the class. |
| Classes have logical existence. | Objects have a physical existence. |
| A class doesn't take any memory spaces when a programmer creates one. | An object takes memory when a programmer creates one. |
| The class has to be declared only once. | Objects can be declared several times depending on the requirement. |

2. What is Encapsulation? Explain with a used case.

Encapsulation is one of the four fundamental OOP concepts. The other three are inheritance, polymorphism, and abstraction. Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.

To achieve encapsulation in Java –

* Declare the variables of a class as private.
* Provide public setter and getter methods to modify and view the variables values.

3. What is Polymorphism? Explain with a used case.

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

Any Java object that can pass more than one IS-A test is considered to be polymorphic. In Java, all Java objects are polymorphic since any object will pass the IS-A test for their own type and for the class Object.

It is important to know that the only possible way to access an object is through a reference variable. A reference variable can be of only one type. Once declared, the type of a reference variable cannot be changed. The reference variable can be reassigned to other objects provided that it is not declared final. The type of the reference variable would determine the methods that it can invoke on the object.

A reference variable can refer to any object of its declared type or any subtype of its declared type. A reference variable can be declared as a class or interface type.

4. Explain Overriding & Overloading and its advantages

Overriding means having two methods with the same method name and parameters (i.e., method signature). One of the methods is in the parent class and the other is in the child class. Overriding allows a child class to provide a specific implementation of a method that is already provided its parent class.

Advantage: -

* Helps in writing generic code based on parent class or interface as object resolution happens at runtime
* Provides multiple implementation of same method and can invoke parent class overridden method using super keyword
* Defines what behaviour a class can have, and implementation of behaviour has been taken care by class which is going to implement.

Overloading in Java is the ability to define more than one method with the same name in a class. The compiler is able to distinguish between the methods because of their method signatures. Overloading in Java creates consistency in the code, which helps eliminate inconsistencies, which could lead to syntax errors. Overloading is also just a convenient way to make the code easier to read through.

Advantage: -

* The main advantage of this is cleanliness of code.
* Method overloading increases the readability of the program.
* Overloaded methods give programmers the flexibility to call a similar method for different types of data.
* Overloading is also used on constructors to create new objects given different amounts of data.
* You must define a return type for each overloaded method. Methods can have different return types.

5. What is Inheritance and different types of inheritance? Explain with a used case.

Inheritance is a mechanism of acquiring the features and behaviours of a class by another class. The class whose members are inherited is called the base class, and the class that inherits those members is called the derived class. Inheritance implements the IS-A relationship.

The types of inheritance are: -

* Single inheritance
* Multi-level inheritance
* Multiple inheritance
* Multipath inheritance
* Hierarchical Inheritance
* Hybrid Inheritance

6. What is an abstract class?

A class which is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

7. What is an interface and how multiple inheritance is achieved with this?

An interface is a reference type in Java. It is similar to class. It is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface. Along with abstract methods, an interface may also contain constants, default methods, static methods, and nested types. Method bodies exist only for default methods and static methods. Writing an interface is similar to writing a class. But a class describes the attributes and behaviours of an object. And an interface contains behaviours that a class implements.

Multiple inheritance by interface occurs if a class implements multiple interfaces or also if an interface itself extends multiple interfaces.

8. What are the access modifiers?

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types of Java access modifiers:

* **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
* **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
* **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
* **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

9. What are the various types of constructors?

There are two type of constructor in Java:

**No-argument constructor**: A constructor that has no parameter is known as default constructor. If we don’t define a constructor in a class, then compiler creates default constructor (with no arguments) for the class. And if we write a constructor with arguments or no-arguments then the compiler does not create a default constructor. Default constructor provides the default values to the object like 0, null, etc. depending on the type.

**Parameterized Constructor**: A constructor that has parameters is known as parameterized constructor. If we want to initialize fields of the class with your own values, then use a parameterized constructor.

10. What is ‘this’ pointer?

'THIS' pointer is a reference variable in Java that refers to the current object.

The various usages of 'THIS' keyword in Java are as follows:

* It can be used to refer instance variable of current class.
* It can be used to invoke or initiate current class constructor.
* It can be passed as an argument in the method call.
* It can be passed as argument in the constructor call.
* It can be used to return the current class instance.

11. What is static and dynamic Binding?

Static Binding is the binding which can be resolved at compile time by compiler is known as static or early binding. Binding of all the static, private and final methods is done at compile-time. In Dynamic binding compiler doesn’t decide the method to be called. Overriding is a perfect example of dynamic binding. In overriding both parent and child classes have same method.

12. How many instances can be created for an abstract class and why?

We can't create an instance of an abstract class because it does not have a complete implementation. The purpose of an abstract class is to function as a base for subclasses. It acts like a template, or an empty or partially empty structure, you should extend it and build on it before you can use it.

13. Which OOPS concept is used as a reuse mechanism and explain with a use case.

Inheritance is the OOPS concept that can be used as reuse mechanism.

14. Please identify one practical scenario for each pillar of OOPs.

**Abstraction**: - One of the most fundamental concept of OOPs is Abstraction. Abstraction is a process where you show only “relevant” data and “hide” unnecessary details of an object from the user. For example: - A car in itself is a well-defined object, which is composed of several other smaller objects like a gearing system, steering mechanism, engine, which are again have their own subsystems. But for human’s car is a one single object, which can be managed by the help of its subsystems, even if their inner details are unknown.

**Encapsulation**: - A example of encapsulation is the class of java.util.Hashtable. User only knows that he can store data in the form of key/value pair in a Hashtable and that he can retrieve that data in the various ways. But the actual implementation like, how and where this data is actually stored, is hidden from the user. User can simply use Hashtable wherever he wants to store Key/Value pairs without bothering about its implementation.

**Inheritance**: - Inheritance is the mechanism by which an object acquires the some/all properties of another object. It supports the concept of hierarchical classification. For example: - Car is a four-wheeler vehicle so assume that we have a class Four-wheeler and a sub class of it named Car. Here, Car acquires the properties of a class Four-wheeler. Other classifications could be a jeep, van etc. Four-wheeler defines a class of vehicles that have four wheels, and specific range of engine power, load carrying capacity etc. Car (termed as a sub-class) acquires these properties from Four-wheeler, and has some specific properties, which are different from other classifications of Four-wheeler, such as luxury, comfort, shape, size, usage etc.

A car can have further classification such as an open car, small car, big car etc, which will acquire the properties from both Four-Wheeler and Car but will still have some specific properties. This way the level of hierarchy can be extended to any level.

**Polymorphism**: - Polymorphism means to process objects differently based on their data type. For example: - A car have a gear transmission system. It has four front gears and one backward gear. When the engine is accelerated then depending upon which gear is engaged different amount power and movement is delivered to the car. The action is same applying gear but based on the type of gear the action behaves differently or you can say that it shows many forms (polymorphism means many forms).