OOPS

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

1. What is Encapsulation? Explain with a used case

Encapsulation is one of the fundamental concepts in object-oriented programming (OOP). It describes the idea of bundling data and methods that work on that data within one unit, e.g., a class in Java. This concept is also often used to hide the internal representation, or state, of an object from the outside. For example for each employee in an employee class, here employee object is the capsule that holds data and methods specific to each employee

1. 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. For instance, let's consider a class Animal and let Cat be a subclass of Animal . So, any cat IS animal.

1. Explain Overriding & Overloading and its advantages

Overloading occurs when two or more methods in one class have the same method name but different parameters. Overriding occurs when two methods have the same method name and parameters. One of the methods is in the parent class, and the other is in the child class. The main advantage of method overriding is that the class can give its own specific implementation to a inherited method without even modifying the parent class 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.

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

Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another. With the use of inheritance the information is made manageable in a hierarchical order. The different types of inheritance are:

* **Single Inheritance**: In single inheritance, a class is allowed to inherit from only one class. i.e. one sub class is inherited by one base class only.

|  |
| --- |
| VEHICLE |

CAR

* **Multilevel Inheritance**: In this type of inheritance, a derived class is created from another derived class.

MOTOR CYCLE

2 WHEELER

VEHICLES

* **Hierarchical Inheritance**: In this type of inheritance, more than one sub class is inherited from a single base class. i.e. more than one derived class is created from a single base class.

BIKE

CAR

BUS

CYCLE

4 WHEELER

2 WHEELER

VEHICLE

1. What is an abstract class?

Abstract class is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class). Abstract method: can only be used in an abstract class, and it does not have a body.

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

The Java programming language supports multiple inheritance of type, which is the ability of a class to implement more than one interface. An object can have multiple types: the type of its own class and the types of all the interfaces that the class implements. This means that if a variable is declared to be the type of an interface, then its value can reference any object that is instantiated from any class that implements the interface.

1. What are the access modifiers?

Access modifiers define the access privileges of classes, interfaces, constructors, methods, and data members. Access modifiers consist of public, private , and protected . If no modifier is present, the default access of package-private is used.

1. What are the various types of constructors?

A constructor is a special method of a class or structure in object-oriented programming that initializes a newly created object of that type. Whenever an object is created, the constructor is called automatically.

* No-argument constructor: A constructor that has no parameter is known as default constructor.
* Parameterized Constructor: A constructor that has parameters is known as parameterized constructor.

1. What is ‘this’ pointer?

The this pointer is a pointer accessible only within the nonstatic member functions of a class , struct , or union type. It points to the object for which the member function is called. Static member functions don't have a this pointer.

1. What is static and dynamic Binding?

Static binding happens at compile-time while dynamic binding happens at runtime. Binding of private, static and final methods always happen at compile time since these methods cannot be overridden. Static binding uses Type information for binding while Dynamic binding uses Objects to resolve binding. Overloaded methods are resolved (deciding which method to be called when there are multiple methods with same name) using static binding while overridden methods using dynamic binding, i.e, at run time.

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

 No you can't, instead you can create instance of all other classes extending that abstract class. Because it's abstract and an object is concrete. An abstract class is sort of like a template, or an empty/partially empty structure, you have to extend it and build on it before you can use it.

1. 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.

For example let us take animal as a class and dogs as its sub class. The dog class acquires every properties of that of animal class and any modifications within the animal class can be reflected within the dog class too.

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

Inheritance: Animal class and a sub class of dogs.

Polymorphism: Adding two numbers and two strings

Encapsulation: TV we only have a power button, It is not required to understand how infra-red waves are getting generated in TV remote control.

Abstraction: Clicking on green button actual send signals to calling person's mobile but we are unaware of how it is doing.

Unit Testing and Junit

1. What is unit testing?

In computer programming, unit testing is a software testing method by which individual units of source code sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine whether they are fit for use.

1. What is the difference between manual testing and automated testing?

In manual testing, a human performs the tests step by step, without test scripts. In automated testing, tests are executed without human assistance, oftentimes via test automation frameworks, along with other tools and software.

1. Is it necessary to write the test case for every logic? If yes, why

Yes, you should unit test everything you can. Doing so creates a legacy for later so changes down the road can be done with peace of mind. It ensures that your code works as expected. It also documents the intended usage of the interfaces.

1. What are the features of JUnit?

JUnit is an open source framework, which is used for writing and running tests. Provides annotations to identify test methods. Provides assertions for testing expected results. Provides test runners for running tests.

1. What are the important JUnit annotations? And its usage in coding

* @BeforeClass – Run once before any of the test methods in the class, public static void
* @AfterClass – Run once after all the tests in the class have been run, public static void
* @Before – Run before @Test, public void
* @After – Run after @Test, public void
* @Test – This is the test method to run, public void

1. What does assert class do ?

Assertions are mainly used to check logically impossible situations. For example, they can be used to check the state a code expects before it starts running or state after it finishes running. Unlike normal exception/error handling, assertions are generally disabled at run-time.

1. What is code coverage?

Code coverage is the percentage of code which is covered by automated tests. Code coverage measurement simply determines which statements in a body of code have been executed through a test run, and which statements have not. In general, a code coverage system collects information about the running program and then combines that with source information to generate a report on the test suite's code coverage.

1. What are the best practices to perform Unit Testing?

* Unit Tests Should Be Trustworthy.
* Unit Tests Should Be Maintainable and Readable.
* Unit Tests Should Verify a Single-Use Case.
* Unit Tests Should Be Isolated.
* Unit Tests Should Be Automated.
* Use a Good Mixture of Unit and Integration Tests.
* Unit Tests Should Be Executed Within an Organized Test Practice.

1. What is mocking ?

Mock testing is an approach to unit testing that lets you make assertions about how the code under test is interacting with other system modules. In mock testing, the dependencies are replaced with objects that simulate the behaviour of the real ones.