**Object Oriented Programming Fundamentals**

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

**Object**

* It is a real world entity that have their own properties and behaviours.
* Object consist of methods and properties
* Object is a physical entity

**Class**

* It is the blueprint from where we can create the objects.
* Class determines how an object will behave and what the object will contain.
* Class is a logical entity

2.What is Encapsulation? Explain with a used case

Encapsulation is one of the fundamental concept in OOPs. It defines the idea of wrapping data and methods together as a single unit. This concept is very useful to hide the internal representations and provides security. 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. S o encapsulation creates private fields and these private fields can’t be accessed by anyone outside the class and also hides the fields within the private class.

3.What is Polymorphism? Explain with a use case

Polymorphism is one of the OOPs feature that allows us to perform a single action in different ways. It describes the concept that different classes can be used with the same interface. Each of these classes can provide its own implementation of the interface. Let’s say we have a class Polygon that have a method draw(). Since it is a generic class we can’t give it a single implementation. It can be Circle, Square, Rectangle, Hexagon, etc. since the class is a polygon and the defined method is draw().

4.Explain Overriding & Overloading and its advantages

**Overloading** is an example for Compile time polymorphism.Overloading occurs when two or more methods in one class have the same method name but different parameters. Overloading increases the readability of the program.Overloading is also used on constructors to create new objects given different amounts of data.

**Overriding** is the example for Runtime polymorphism. 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.

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

Inheritance is one of the core concept of OOPs in which one class acquires the property of another class. Like a child inherits the traits of his parents. Using inheritance, we can reuse the methods of an existing class. Hence, inheritance promotes reusability.

Types pf inheritance

* Single Inheritance

When a class extends(inherit) another one class only then we  call it a single inheritance

* Multiple Inheritance

When one class extends(inherits) more than one class or multiple classes, we call it a multiple inheritance

* Multilevel Inheritance

Where one can extends(inherit) from a derived class, thereby making this derived class the base class for the new class, we call it a Multilevel inheritance.

* Hierarchical Inheritance

When one class is inherited by many**sub classes or one base class is inherited by many sub classes we call it a Hierarchical inheritance**

* Hybrid Inheritance

Hybrid inheritance is a combination of**Single** and **Multiple**inheritance.

6.What is an abstract class?

Abstract class is a class declared with abstraction. Abstraction is the process of hiding implementation details and showing only the functionality to the user.

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

An **interface** is an abstract type which stores methods and helps other classes to implement the behaviour of the method stored by the interface. We need interfaces to achieve multiple inheritance. If there is a superclass inherited by two classes class A and class B and if we try to inherit class A and class B to a new class C, we need interface. Interface stores the methods and behaviours of class A and class B and using interface, class C can inherit all those methods.

8.What are the access modifiers?

Access modifiers set access levels for classes, variables, methods, and constructors. They restrict the accessibility of classes, variable, methods, and constructors.

9.What are the various types of constructors?

Constructor is a block of code and it is called when an instance of a class is created. There are two type of constructors:

* Default constructor – A constructor which doesn’t have any parameters.
* Parameterized constructor – A constructor which has a specific number of parameters.

10.What is ‘this’ pointer?

It is a reference variable that refers to the current object in a method or constructor. It can also be passed as an argument in method and constructor.

11.What is static and dynamic Binding?

Static binding refers to the execution of a program where type of object is known at compile time. That is, when compiler executes the code it know the type of object or class to which object belongs. Static binding uses type of class for binding

Dynamic binding refers to the execution of program where type of object is determined at runtime. Dynamic binding uses type of object for binding because object is created during runtime and as a result, dynamic binding is slower compared to static binding.

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

We can’t create instances for an abstract class. 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, we have to extend it and build on it before we 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 a reuse mechanism. In inheritance, one class acquires the property of another class. Like a child inherits the traits of his parents. Using inheritance, we can reuse the methods of an existing class. For example, consider a class Birds with list of some birds. We can create a subclass Flying birds that inherits from the class Birds.

Here, data is reused from parent class Birds. We can only use this reuse mechanism in Inheritance.

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

There are many practical scenarios where we can see Object Oriented Programming concept. To make it simple consider the following example.

Let’s take a mobile phone. We can consider mobile phone as an Object. There are many mobile manufacturing companies. So each company can be an Object and each object can be different based on their characteristics (For example, iPhone, Samsung).

A Class is a blueprint of how the object should be represented. A Mobile Class consist of name, attributes, and operations. In the above example we can take IMEI Number, Processor, etc as attributes and Dial, Receive, Send Message as Operations.

Abstraction expose limited data functionality of objects publicly and hides the actual implementations. Consider above example, Mobile Class and Objects like Samsung and iPhone

Dialing a number call some methods internally as a result, we can see dialed numbers on screen.

Encapsulation is defined as the process of enclosing one or more details from outside world. Both Abstraction & Encapsulation works hand in hand because Abstraction says what details to be made visible and Encapsulation provides the level of access right to that visible details. That is, It implements the desired level of abstraction.

Consider Bluetooth n Mobile. We can enable Bluetooth and connect with any other device or another Mobile Phone, But we are not able to access the features of that mobile like dialing number, accessing inbox etc. Because Bluetooth have some level of abstraction.

Polymorphism can be defined as the ability of using the same name for doing different things.

Consider the example Mobile camera, it is having a functionality of Capture(). Now same mobile is having Panorama mode available in camera, So functionality is to capture photo but it have different modes to capture photos.

Inheritance is the ability to extend the functionality from base entity to the new entity. This will help us to reuse the functionality which is already defined before and extend into a new entity.

Consider the above example Mobile Class, basic Mobile functionality is to send a message, dial and receive a call. So the brands of mobile is using this basic functionality by extending the mobile class functionality and adding their own new features to their respective brand.

Now a Mobile phone have many features but every Mobile have these basic functionalities (send a message, dial and receive a call) which are inherited from his previous generations.

**Unit Testing & Junit**

1.What is unit testing?

Unit Testing is a type of software testing where software is divided into individual components or units and these individual units are tested.

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

Manual testing

* Manual testing is testing of the software where tests are executed manually by a Tester alone.
* Manual testing process is not accurate because there is a possibility of human errors.
* Manual testing is time consuming
* Possible without programming knowledge.
* Random testing is possible

Automated Testing

* Testing is done with the use of scripts, code and automation tools(computer) by a tester.
* This testing is more accurate compared to manual testing because it is code and script based.
* Testing is very fast
* Testing is not possible without programming knowledge
* Doesn’t allow random testing

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

It is necessary to write test cases. The key purpose of a test case is to ensure if different features within an application are working as expected. It helps tester to ensure it is working as per the expectations of the end users.

4.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 declarations for testing expected results.
* Provides test runners for running tests.
* JUnit tests allow you to write codes faster, which increases quality.
* JUnit is elegantly simple. It is less complex and takes less time.
* JUnit tests can be run automatically and they check their own results and provide immediate feedback

5.What are the important Junit annotations? And its usage in coding

* @Test – This annotation indicates that public void method to which it is attached can be executed as a test Case
* @Before - This annotation is used if you want to execute some statement such as preconditions before each test case.
* @BeforeClass - This annotation is used if you want to execute some statements before all the test cases for example, test connection must be executed before all the test cases
* @After - This annotation can be used if you want to execute some statements after each Test Case for example, resetting variables, deleting temporary files, variables, etc.
* @AfterClass - This annotation can be used if you want to execute some statements after all test cases for e.g. Releasing resources after executing all test cases
* @Ignore - This annotation can be used if you want to ignore some statements during test execution for example, disabling some test cases during test execution.

6.What does Assert class?

Assert class is a class provided by Junit, which provides a bunch of assertion methods useful in writing test cases and to detect test failure.

7.What is Code Coverage?

Code coverage is the percentage of code which is covered by automated tests. Code coverage  determines which statements in a body of code have been executed through a test run, and which statements have not executed.

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

* Unit Testing should be trustworthy.
* Unit Testing should be Maintainable and Readable.
* Unit Testing should verify a single-use case.
* Unit Testing should be Isolated.
* Unit Testing should be Automated.
* Use a good Mixture of Unit and Integration tests.
* Unit Testing should be executed Within an organized test practice

9.What is Mocking?

Mocking means creating a fake version of an external or internal service that can stand in for the real one, helping our tests run more quickly and more reliably.