**Object Oriented Programming Fundamentals**

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

A class is an entity that determines how an object will behave and what the object will contain

Object is a self-contained component that consists of methods and properties to make a data useful

Class

A class is a template for creating objects in program.

You can declare class only once

A class is a logical entity

Object

The object is an instance of a class

Object is a physical entity

Object allocates memory space whenever they are created

can create more than one object using a class

1. What is Encapsulation? Explain with a used case

**Encapsulation**is defined as wrapping up of data and information under a single unit binding together the data and the functions that manipulates them.

Eg - when a car starts there are certain internal events happened, like spark gets ignited, tachometer shows its changes, fuel is pumped, and the end user won’t know the internal events which happened. Because the driver accessed the car through specific interface, and the internals events are encapsulated. The driver don’t need to know about the mechanics happened inside the car but the car will be moving

1. What is Polymorphism? Explain with a used case

it is the ability of an object to take many forms , is the method in an object-oriented programming language that does different things depending on the class of the object which calls it

Eg - I can behave as what situation demands. Can act as a student when I am in school, can act as a customer while I am in a supermarket, can act as a passenger while traveling in a bus .That is more than one function with same name with different working is called as a polymorphism

1. 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

Is the mechanism in which one class inherits the property of other class

Eg - consider father and son relationship, son can inherit all the fathers’ property but father

Cant

Types of inheritance

Single Inheritance

When a class extends (inherit) another one class only then

Multiple Inheritances

When one class extends (inherits) more than one class or multiple classes,

Multilevel Inheritance

Where one can extends (inherit) from a derived class, thereby making this derived class the base class for the new class,

Hierarchical Inheritance

When one class is inherited by many**sub classes or one base class is inherited by many sub classes**

Hybrid Inheritance

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

1. What is an abstract class

An abstract class is a class which cannot be instantiated. Creation of an object is not possible with abstract class , but it can be inherited. An abstract class can contain only Abstract method

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

An interface stores methods and helps other classes to implement the behavior of the method stored by the interface

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 behaviors of class A and class B and using interface, class C can inherit all those methods

1. 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.

1. What are the various types of constructors

A constructor is a member function of a class which initializes objects of a class. Constructor is automatically called when object is (instance of class) create. It is special member function of the class.

Default constructor – A constructor which doesn’t have any parameters.

Parameterized constructor – A constructor which has a specific number of parameters.

1. 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.

1. What is static and dynamic Binding

Static binding - compile time

Dynamic binding - runtime

Static binding - refers to the execution of a program where type of object is determined/known at compile time . When compiler executes the code it know the type of object or class to which object belongs.

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

1. 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

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 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

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

Think there are two vehicles one is a sedan, and the other one is a suv and can name both these vehicles as a single name that is car

When we talk this car example into oops

Class and object: - A Class is a blueprint of how the object should be represented. In this car example there are two vehicles one is sedan and the other one is a Suv the functionality of both these vehicles are different but we can consider both as a car

So car will be the main class consist of both the vehicles and the subclass will be sedan and suv which we can call it as an object

Suv (object)

Sedan (object)

Abstraction:- expose limited data functionality of objects publicly and hides the actual implementations

Eg: -When we drives a car we will be only giving the throttle combustion happens and car starts to move, we will be only feeling the car movement not the internal events happened

**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.

Eg: - when a car starts there are certain internal events happened, like spark gets ignited, tachometer shows its changes, fuel is pumped, and the end user won’t know the internal events which happened. Because the driver accessed the car through specific interface, and the internals events are encapsulated. The driver don’t need to know about the mechanics happened inside the car but the car will be moving

**Polymorphism:-** can be defined as the ability of using the same name for doing different things

it is the ability of an object to take many forms

Eg: - Consider the Suv actually it’s a car and the main functionality is to travel through road , but we can use this vehicle in many terrains ,like in mud , to climb hill ,also we can tow other vehicles . That is same vehicle but different purpose

**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

Eg : -When a car is outdated manufactures will make new cars based on the previous model updated model will have all the properties of the outdated car but outdated car won’t be having all the functions which is in the updated car

**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.

1. 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

1. 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.

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 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

1. 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.

1. 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.

1. 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.

1. 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

1. 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.