**Object Oriented Programming Fundamentals**

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 as a variable of the class.

1. What is Encapsulation? Explain with a used case

Encapsulation is an Object Oriented Programming concept that binds together the data and functions that manipulate the data, and that keeps both safe from outside interference and misuse. Data encapsulation led to the important OOP concept of data hiding.

The best example of encapsulation could be a calculator. We understand its interface from first sight and we don't have to know how it works inside. We know that we can press 2+2 then = and see the result on display.

1. What is Polymorphism? Explain with a used case

Polymorphism is an object-oriented programming concept that allows the object to take multiple forms.

Eg: add(), add(int a, int b)

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 (Compile Time polymorphism). An advantage is that if we must perform only one operation such as addition, having same name of the methods like ‘add’ increases the readability of the program.

Overriding allows a child class to provide a specific implementation of a method that is already provided its parent class (Runtime Polymorphism). Advantage is that this provides multiple implementation of the same method and can invoke parent class overridden method using super keyword.

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

Inheritance is an OOPs concept where one class acquires all the properties of their parent class. There are mainly 5 types of inheritance: single inheritance, multiple inheritance, multilevel inheritance, hierarchical inheritance and hybrid inheritance.

Eg: child class inheriting the traits of its parent class.

1. What is an abstract class?

An abstract class is a class that is declared abstract. Abstract class should contain one abstract method and it does not support multiple inheritance.

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

An interface contains variables and methods like a class but the methods in an interface are abstract by default unlike a class. Multiple inheritance by interface occurs if a class implements multiple interfaces or also if an interface itself extends multiple interfaces.

1. What are the access modifiers?

Access modifiers are keywords in object-oriented languages that set the accessibility of classes and methods.

1. What are the various types of constructors?

There are three types of constructors: Default, No-argument constructor and Parameterized.

1. What is ‘this’ pointer?

THIS works as a reference to the current object, whose method or constructor is being invoked.

1. What is static and dynamic Binding?

Static binding is a binding in which name can be associated with the class during compilation time and Dynamic binding is a binding in which name can be associated with the class during execution time.

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

None, because 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.

1. Which OOPS concept is used as a reuse mechanism and explain with a use case Inheritance is the concept that can be used as reuse mechanism. For e.g.: when a child class extends the properties or methods of a parent class using the inheritance feature, the same need not be defined again in the child class.
2. Please identify one practical scenario for each pillar of OOPs.

* Abstraction- Clicking on green button actual send signals to calling person's mobile but we are unaware of how it is doing.
* Encapsulation- When we switch on a Bluetooth, I am able to connect to another mobile or bluetooth enabled devices but I'm not able to access the other mobile features like dialing a number, accessing inbox etc.
* Polymorphism- Samsung mobile has a 5MP camera available i.e. – it is having a functionality of CameraClick(). Now same mobile is having Panorama mode available in camera, so functionality would be same but with mode. This type is said to be Compile time polymorphism.
* Inheritance- 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.

**Unit Testing & JUnit**

1. What is unit testing?

UNIT TESTING is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

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

In manual testing, test cases are executed manually (by a human) without any support from tools or scripts. But with automated testing, test cases are executed with the assistance of tools, scripts, and software.

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

Yes, you should write a test case for 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.

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.
* 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. There's no need to manually comb through a report of test results.
* JUnit tests can be organized into test suites containing test cases and even other test suites.
* JUnit shows test progress in a bar that is green if the test is running smoothly, and it turns red when a test fails.

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

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Annotations** | **Description** |
| 1. | @Test | This annotation is a replacement of org.junit.TestCase which indicates that public void method to which it is attached can be executed as a test Case. |
| 2. | @Before | This annotation is used if you want to execute some statement such as preconditions before each test case. |
| 3. | @BeforeClass | This annotation is used if you want to execute some statements before all the test cases for e.g. test connection must be executed before all the test cases. |
| 4. | @After | This annotation can be used if you want to execute some statements after each[Test Case](https://www.guru99.com/test-case.html)for e.g resetting variables, deleting temporary files ,variables, etc. |
| 5. | @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. |
| 6. | @Ignores | This annotation can be used if you want to ignore some statements during test execution for e.g. disabling some test cases during test execution. |
| 7. | @Test(timeout=500) | This annotation can be used if you want to set some timeout during test execution for e.g. if you are working under some SLA (Service level agreement), and tests need to be completed within some specified time. |
| 8. | @Test(expected=IllegalArgumentException.class) | This annotation can be used if you want to handle some exception during test execution. For, e.g., if you want to check whether a particular method is throwing specified exception or not. |

1. What does Assert class?

Assert is a method useful in determining Pass or Fail status of a test case, The assert methods are provided by the class org. Assert which extends java. lang. Object class.

1. What is Code Coverage?

Code coverage is a measurement of how many lines/blocks/arcs of your code are executed while the automated tests are running. Code coverage is collected by using a specialized tool to instrument the binaries to add tracing calls and run a full set of automated tests against the instrumented product.

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.