**Object Oriented Programming Fundamentals**

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

A class is a template for objects or object is an instance of a class.class is used to define an object. Object can be related to real world objects. Object can be pen , pencil , mobile phone , person etc, while class is a group of similar object .Object can be created may times , but a class be created only once.

1. **What is Encapsulation? Explain with a used case?**

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

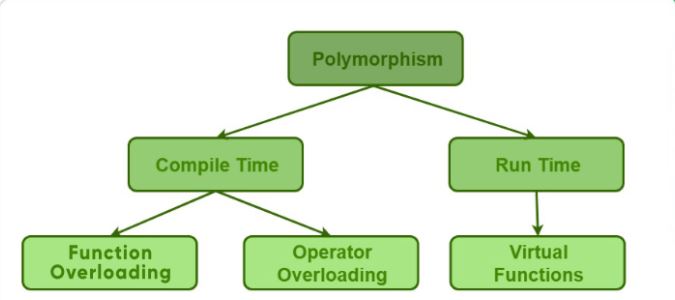
Consider a reallife example of encapsulation, in a company there are different sections like the accounts section, finance section, sales section etc. The finance section handles all the financial transactions and keep records of all the data related to finance. Similarly the sales section handles all the sales related activities and keep records of all the sales. Now there may arise a situation when for some reason an official from finance section needs all the data about sales in a particular month. In this case, he is not allowed to directly access the data of sales section. He will first have to contact some other officer in the sales section and then request him to give the particular data. This is what encapsulation is. Here the data of sales section and the employees that can manipulate them are wrapped under a single name “sales section”.

1. **What is Polymorphism? Explain with a used case?**

It means the ability to behave differently in different inputs.ie the ability to perform different ways .polymorphism allows us to define on interface and different implementations.

In real life :  Like a man at the same time is a father, a husband, an employee. So the same person posses different behavior in different situations

2 type Polymorphism



**Function Overloading**: When there are multiple functions with same name but different parameters then these functions are said to be overloaded. Functions can be overloaded by change in number of arguments or/and change in type of arguments.

**Operator Overloading**:we an define operator to function differently depending on the inputs . we can define “+” to add complex numbers ,

[**Function overriding**](https://www.geeksforgeeks.org/override-keyword-c/) :on the other hand occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be **overridden**.

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.

* The main advantage of this is cleanlinessof code.
* Method overloading increases thereadability of the program.
* Overloaded methods give programmers theflexibility to call a similar method for different types of data

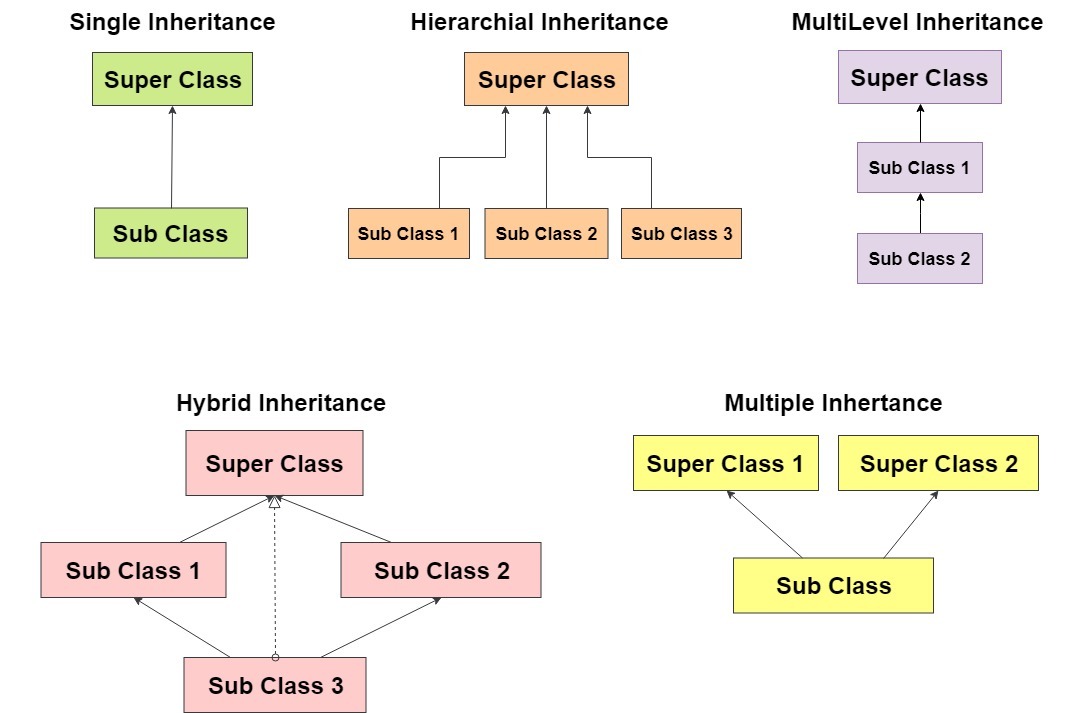
**Overriding** occurs when two methods have the same method name and parameters. One of the methods is in the parent class, and the other is in the child class. Overriding allows a child class to provide the specific implementation of a method that is already present in its parent class

* 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 behavior a class can have and implementation of behavior has been taken care by class which is going to implement.

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

Inheritance is a mechanism of acquiring the features and behaviors of a class from 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.

Types of inheritance:



The idea behind inheritance is that you can create new [classes](https://www.javatpoint.com/object-and-class-in-java) that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

* For Method Overriding (so runtime polymorphism can be achieved).
* For Code Reusability.

1. **What is an abstract class?**

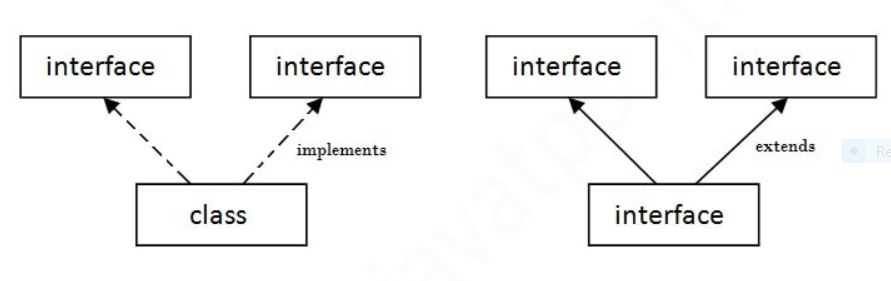
Declaring a class as abstract means that it cannot be directly instantiated, which means that an object cannot be created from it. That protects the code from being used incorrectly. Abstract classes require subclasses to further define attributes necessary for individual instantiation. The purpose of an abstract class is to define some common behavior that can be inherited by multiple subclasses, without implementing the entire class.

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

Like a class, an interface can have methods and variables, but the methods declared in an interface are by default abstract (only method signature, no body)

* Interfaces specify what a class must do and not how. It is the blueprint of the class.
* An Interface is about capabilities like a Player may be an interface and any class implementing Player must be able to (or must implement) move(). So it specifies a set of methods that the class has to implement.

Multiple inheritance using interface can be achieved such that ,If a class implements multiple interfaces, or an interface extends multiple interfaces, it is known as multiple inheritance.



1. **What are the access modifiers?**

Access modifiers are keywords in object-oriented languages that set the accessibility of classes, methods, and other members. Access modifiers are a specific part of programming language syntax used to facilitate the encapsulation of components. There are mainly 3 access specifiers :**Public , Private , Public.**

Public :

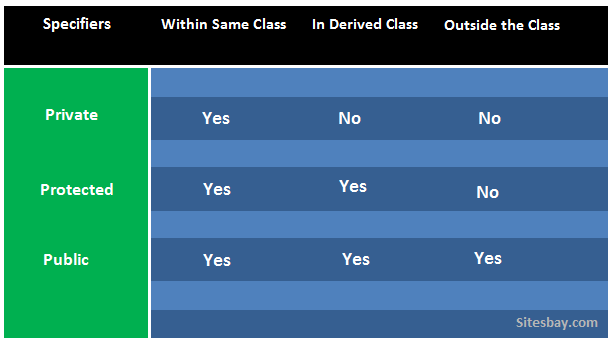
Those structures declared under is available to access within the same class ,in the derived class and outside the class.

Private :

Those structures declared under private is only accessible with in the class only , not accessible to the derived class and outside the class.

Protected:

Those structures defined under protected is accessible with in the same class and to the derived class also , but is accessible outside the class.



1. **What are the various types of constructors?**

A constructor is a member function of a class which initializes objects of a class.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. It name of the constructor is same as that of the class. It used to initialize the object while creating an object of that class. Constructor is a block of code that initializes the newly created object.

Type of constructors:

* Parameterized constructors
* Default constructors
* Copy constructors

**Parameterized constructors:**

Constructors that can take at least one argument are termed as parameterized constructors. When an object is declared in a parameterized constructor, the initial values have to be passed as arguments to the constructor function.

**Default constructors:**

A default constructor is a constructor that either has no parameters, or if it has parameters, all the parameters have default values.

**Copy constructors:**

A copy constructor is a member function that initializes an object using another object of the same class.

1. **What is ‘this’ pointer?**

The this pointer is a constant pointer that holds the memory address of the current object

1. **What is static and dynamic Binding?**

static binding refers to the execution of a program where type of object is determined/known at compile time i.e ,when compiler executes the code it know the type of object or class to which object belongs. While in case of dynamic binding the type of object is determined at runtime.

Static Binding: 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.

Dynamic Binding: 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 .

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

We cannot create an instance of an abstract class, The purpose of an abstract class is to define some common behavior that can be inherited by multiple subclasses, without implementing the entire class. So abstract class cannot instantiated.

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.

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

* Abstraction : Abstraction is the process of showing only essential/necessary features of an entity/object to the outside world and hide the other irrelevant information. For example to open your TV we only have a power button, It is not required to understand how infra-red waves are getting generated in TV remote control.
* Encapsulation : Encapsulation means wrapping up data and member function (Method) together into a single unit. If combinations of medicine are variables and methods then the capsule will act as a class and the whole process is called Encapsulation.
* Inheritance : The ability of creating a new class from an existing class. Inheritance is when an object acquires the property of another object. Inheritance allows a class (subclass) to acquire the properties and behavior of another class (super-class). It helps to reuse, customize and enhance the existing code. Inheritance : The ability of creating a new class from an existing class. Inheritance is when an object acquires the property of another object. Inheritance allows a class (subclass) to acquire the properties and behavior of another class (super-class). It helps to reuse, customize and enhance the existing code
* polymorphism : It means the ability to behave differently in different inputs.ie the ability to perform different ways .polymorphism allows us to define on interface and different implementations.In real life :  Like a man at the same time is a father, a husband, an employee.

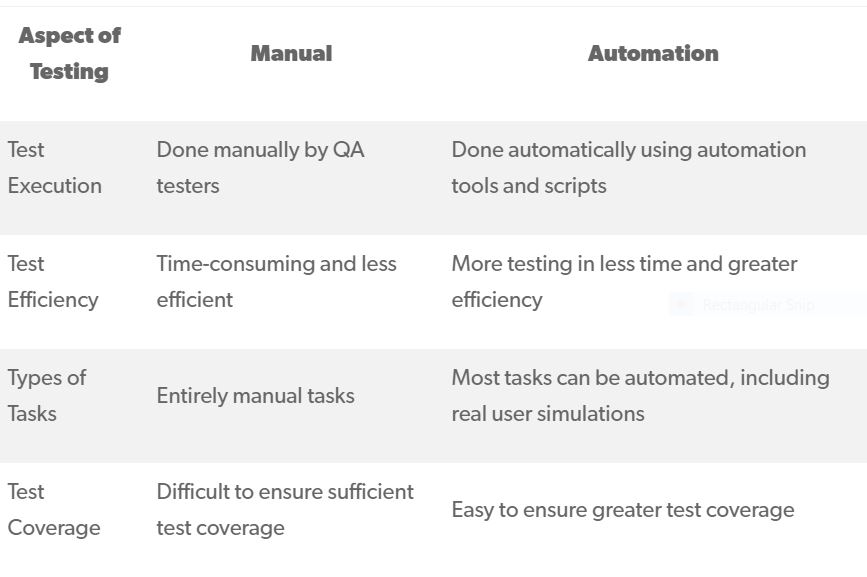
**Unit Testing & Junit**

1. **What is unit testing?**

UNIT TESTING is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object. In unit testing the code is tested in the developer side , so testing the individual modules helps in determining the bugs in the developer side it self .

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

There are some major differences in automated testing vs. manual 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 , in my opinion it necessary to write test cases to for each logic , because then only we will be able to get more code coverage , that makes our code almost bug free and error free and make the code more stable.

1. **What are the features of JUnit?**

JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development. and is a family of unit testing frameworks collectively known as xUnit.

Features of Junit testing.

* + 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**

Junit annotations is a special form of syntactic meta-data that can be added to Java source code for better code readability and structure. Variables, parameters, packages, methods and classes can be annotated. Annotations were introduced in Junit4, which makes Java code more readable and simple.

Common junit annotations are @Test,@Before,@BeforeClass ,@After,@AfterClass etc.

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.junit.Assert which extends java.lang.Object class.

Common assert methods are

* **Boolean**

If you want to test the boolean conditions (true or false), you can use following assert methods

Here the condition is a boolean value.

**assertTrue(condition)**

**assertFalse(condition)**

* **Null object**

If you want to check the initial value of an object/variable, you have the following methods:

**assertNull(object)**

**assertNotNull(object)**

1. **What is Code Coverage?**

Code coverage is the percentage of code which is covered by tests. Thus seeing code coverage we can understand how much of your code has been tested .

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
* Unit Tests Should Be Executed Within an Organized Test Practice
* Every bug you fix should have a test that verifies the bug is fixed

1. **What is Mocking?**

In object-oriented programming, mock objects are simulated objects that mimic the behavior of real objects in controlled ways, most often as part of a software testing initiative. A programmer typically creates a mock object to test the behavior of some other object, in much the same way that a car designer uses a crash test dummy to simulate the dynamic behavior of a human in vehicle impacts