C++ NOTE 1: OBJECT ORIENTED PROGRAMMING

- Object oriented programming is nothing but a programming technique or a way we write code. Before understanding what Object-Oriented Programming (OOP) does and how it came to be, we need to understand what came before OOP.

  Before OOP became a popular way to write code the most used programming technique was procedural programming and functional programming.

- In procedural programming there was a bunch of sequential instructions that the computer needed to execute in order to perform a particular task for eg. When you upload a photo to Instagram you first click the photo, edit it, add a song to it and then post it to Instagram, this signifies a number of steps done one after another and not jumbled execution, you cannot edit a photo before you clicked it.
- An extension of procedural programming was the introduction of functional programming where we use functions to divide the tasks and make them available to us no matter how many times we need them. For eg. If you have an Instagram account you don't just post one photo you post 100s, so while posting each photo you don't want to write the instructions again and again, you just want to write it once and just use that feature as many times you want.

  Example Code: https://drive.google.com/file/d/1LegI8tErMSPEPow7qdYeNGib4-cGCZy-/view?usp=sharing

  Comment out parts of the code to understand the functioning

  Some programming languages that use procedural or functional programming techniques are COBOL (earlier versions), PASCAL, C, etc.

  Now in programming languages such as C, the data stored in say a Global variable (These are declared before the main function) is freely accessible to all the available functions whatever they might be.

- For example, every Instagram photo you post has a caption, but you don't want one photo to be able to have the caption of another photo. This was one of the major problems related to functional or procedural programming. Though these concepts are still used within the programming world the OOP technique is the most famous one as of now.

Now as we understand what came before OOP, we step into the world of OOP and understand what it is.

- The main "Hero" of OOP are, as the name suggests, Objects. What are objects?
- Objects in OOP can correspond to real world objects. In a simple way, they are made up of certain characteristics and behaviours. A human being can be an object, a light bulb can be an object and any other real-world entity imaginable.
- Continuing our example, let us say you post a photo on Insta, that photo will have several characteristics or **attributes** (as we call them in programming) such as Date uploaded on, caption, no. of likes, music attached, people tagged, location etc. and also will have certain behaviours or **Functions** such as double tapping it to like, click that tagged person to go to their account etc.

  Thus, the photo you post becomes an object and several other photos will have the same attributes and functions, just having different values.

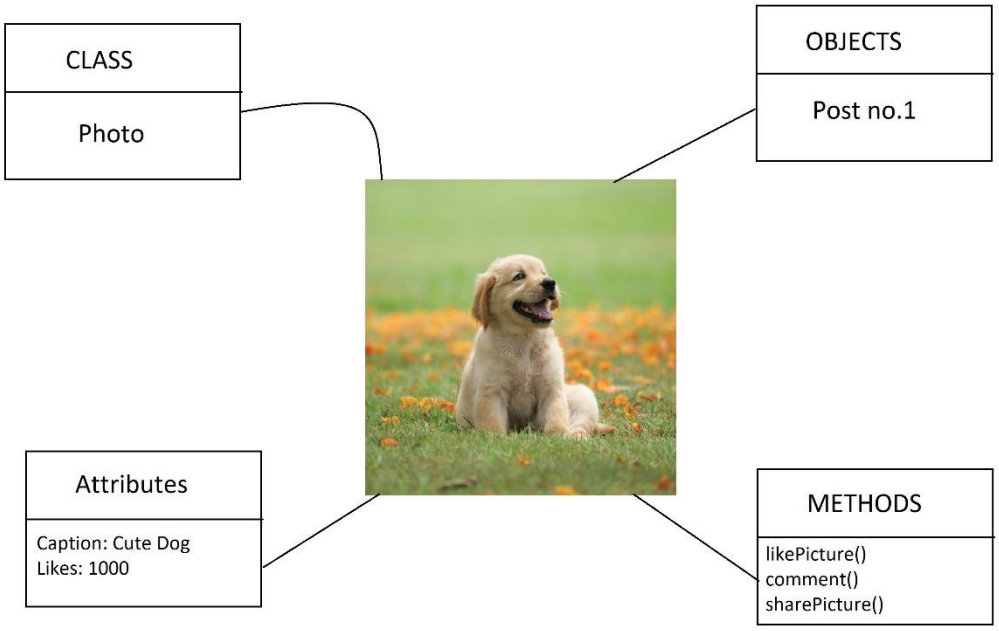
- Now, you would want to hide the values of the attributes of one photo from another even though the attribute itself is the same. And also, if I double tap on one photo the like count of that photo must go up and not of the other one.

  So, by this we understand that, the value of the characteristics within a particular object is only changeable by the function or behaviour that is related to the object and not by any other external function.

- As a result of the functionalities the code written in OOP design becomes more flexible, modular and abstract. This is usually helpful when you write large programs.

Before we move on to the features of OOP, we need to understand the parts of it:

- The OOP design paradigm is based on several parts, some of which are:
  i) **Classes**: As we understood objects, Classes are nothing but blueprint of those objects. We have photos of Insta as objects, but determining what characteristics they should possess and what functions they should provide are defined and determined by the classes. The classes have **no relation** to the value of characteristics.
  These are user defined data types and encapsulate various other data types within it according to the needs of the program.
  Assignment: Find and Understand the Differences between a Structure and a Class in C++.
  ii) **Objects:** These are instances of the classes, meaning, you can post 100 or 1000 photos based on the blueprint provided by the class but the data determining the photo can be different from the rest of the photos.
  iii) **Attributes:** These are characteristics of a particular object that store the value determining the state in which the object is. For example, no. of likes is a Attribute that determines at the point of time how many likes are registered to that particular photo.
  iv) **Functions:** These describe the behaviour of an object that is, these perform operations on the values stored in the Attributes. Double tapping increases the value of the no. of likes characteristic by +1.
- For example: You post a photo on your Insta page:



Features or Pillars of OOP:

- **Data Hiding:** Data hiding is an object-oriented programming technique for protecting the data within a class from unwanted access and preventing unneeded intrusion from outside the class. It helps hide internal object details, i.e., data members within the class, to prevent its direct access from outside the class**.**

- **Abstraction:** The meaning of abstraction is pretty simple to understand, you do a thing and figure out the result but don't want to share to your friends how you did it. You are "abstracting" the inner workings of the work that you did.
  Definition: Data abstraction is the process of exposing to the outside world only the information that is absolutely necessary while concealing implementation or background information.

- **Data Encapsulation:** We have understood what are classes, which are the key component when it comes to data encapsulation. Think of a medical capsule that holds tiny medicinal components that can be of various types and contain different properties.
  Definition: Data encapsulation is one of the key features of object-oriented programming. It is the process of grouping or bundling the data members and member functions into a single unit known as a class.

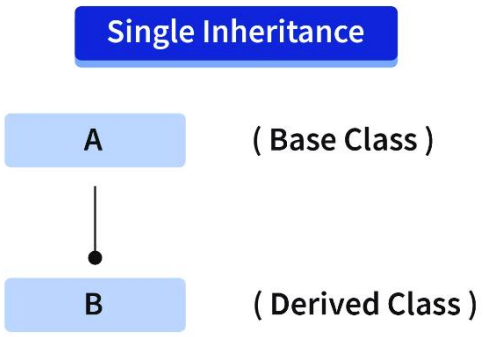
- **Polymorphism:** The basic idea behind polymorphism is that an object or function can behave differently in different contexts. Think of your friend who is the most naughtiest person you have ever met in front of all your friends but in front of their parents they act like the most innocent living organism in the planet.
  Types:
    1. Compile Time polymorphism: Compile-time polymorphism is done by overloading an operator or function. Overloaded functions are called by comparing the data types and number of parameters whose information is available to the compiler at compile time, thus this is known as Compile time polymorphism.
       Types:
       a. Operator overloading: When an operator is updated to be used for user-defined data types (objects etc.), this is known as operator overloading.

       b. **Function overloading**: When we have two functions with the same name but different parameters, different functions are called depending on the number and data types of parameters. This is known as function overloading.

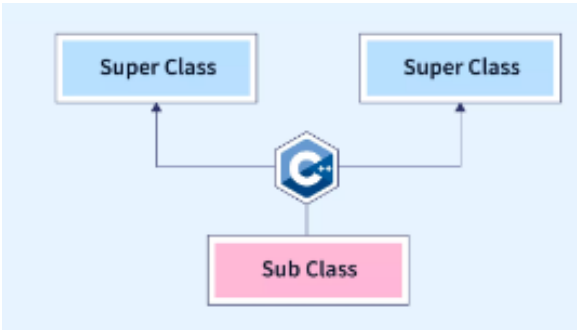
    2. Runtime polymorphism: Runtime polymorphism occurs when functions are resolved at runtime rather than compile time when a call to an overridden method is resolved dynamically at runtime rather than compile time.
- **Inheritance:** When one object acquires all the properties and behaviours of parent object i.e. known as inheritance. It provides the ability to reuse code. While inheriting properties classes are divided into two types:
    a. Sub-class or Derived Class: These are classes that inherit the properties. You can also call them as child classes.
    b. Super-class or Base Class: These are classes from which the Derived classes receive properties from. These can also be called as parent classes.

Various types of Inheritance:

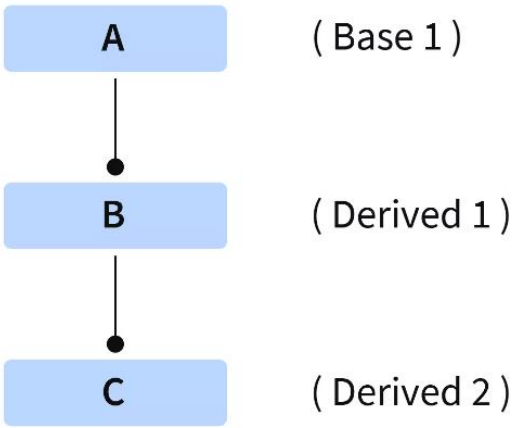
a. Single inheritance: Simple to understand one base class and one derived class.



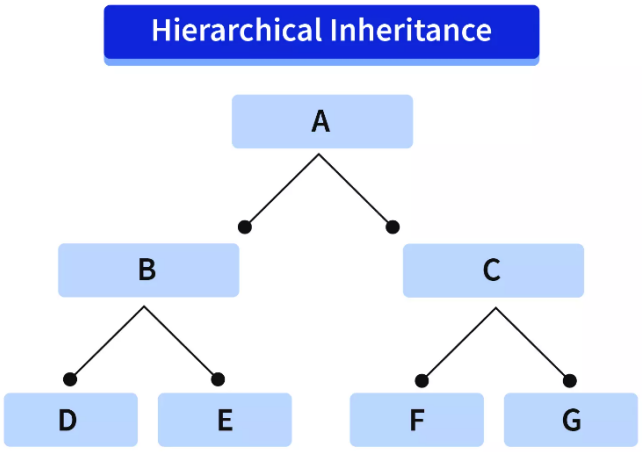
b. Multiple Inheritance: When a derived class(child class) inherits more than one base class(parent class), it is called multiple inheritance.



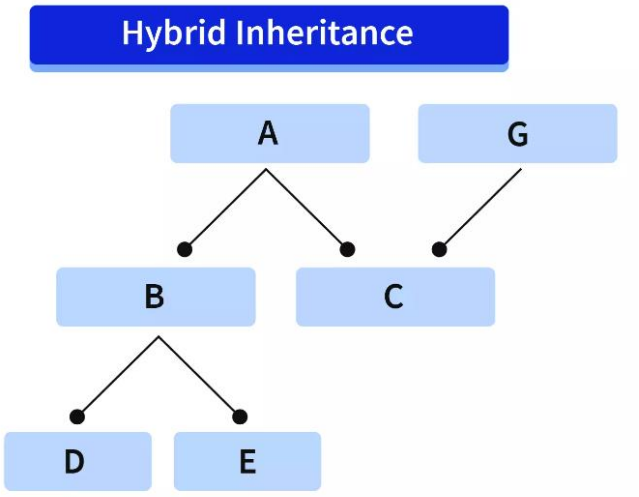
c. **Multilevel Inheritance:** When a derived(child) class inherits the base class and acts as the base class(parent class) to the other class, it is called Multilevel Inheritance. There can be any number of levels i.e any number of derived classes in multilevel inheritance.



d. **Hierarchical Inheritance**: When more than one class is inherited from a single base class, it is called Hierarchical Inheritance.



e. **Hybrid Inheritance:** It is a combination of one or more types of inheritance.



Assignment: Find the Advantages of OOP over other Programming methodologies such as Procedural and Function oriented/Functional Programming.

NOTE: Access Specifiers or Modifiers:

- `public` - members are accessible from outside the class
- `private` - members cannot be accessed (or viewed) from outside the class
- `protected` - members cannot be accessed from outside the class, however, they can be accessed in inherited classes.