Object Oriented Programming

Object oriented programming is a paradigm based on the concept of objects which can contain data, in the form of fields, and code, in the form of procedure.

* **Class and Object**

**Classes**

A class isn terms of OOP is a blueprint for creating objects. A class defines a datatype by building data and methods that work on the data into one single unit.

**Object**

Object is a instance of a class. It is a specific implementation of the class that can be used to perform the operations defined by the class.

* **Inheritance**

In programming, inheritance refers to passing down characteristics from a parent to a child so that a new piece of code can reuse and build upon the features of existing one.

It is a mechanism by which one class can inherit the attributes and methods of another class. we are reusing and extending existing code.

The class which inherits properties and method is called the child/sub class and the class whose properties and methods are inherited are called the parent/base class.

JavaScript uses prototypal inheritance, don’t classical inheritance like Java / C++. Typescript uses class based inheritance which is simply the syntactic sugar of prototypal inheritance.

Typescript supports only single inheritance and multilevel inheritance. In typescript a class inherits another class using extends keyword.

**Single Inheritance**

In single inheritance, the properties and behavior of the base class can be inherited into at most one derived class. it used to add new functionality to the already implemented class.

**Multilevel inheritance**

In multilevel inheritance, the derived class acts as the base class for another derived class. The newly created derived class acquires the properties and behavior of other base classes.

Access modifiers

Access modifiers are keywords used to set the accessibility of classes, methods, and other members. They control the visibility and accessibility of these elements within and outside the class.

* Public
* Private
* Protected

**Abstraction**

Abstraction in oops is used to hide unnecessary information and display only necessary information to the users interacting. It is essential to represent real world objects in a simplified manner for users to interact easily.

Type of abstraction in oops

There are two types of abstraction in object oriented programming

* **Data abstraction in oops**

Data abstraction is the most straightforward implementation of abstraction. When we dealing with a lot of complex processes, we hide unnecessary data and display only the information necessary for the users. Classes define data members as attributes and functions as the method that is used to perform the required operation on data.

User interact with the system through method calls present in the classes. It keeps the complex operations hidden from the users during implementation.

* **process abstraction in oops**

when we hide the internal implementation and do not disclose all the details about a method or function to the users, this is known as process abstraction. In this type of abstraction, instead of dealing with data, we deal with their process.

**Advantage of abstraction in oops**

* + Abstraction is used to present the necessary information to our users and simplify their interaction with our application.
  + One can group many related classes using abstraction.
  + Abstraction in oops helps users avoid writing low level code.
  + The user does not need to worry about the internal implementation of the application.
  + It helps reduce the complexity of iteracting with the system and also increases its readability.
  + It helps developers increase their development speed and ease.

**Encapsulation**

Encapsulation is a fundamental concept in object oriented programming that combines the data and methods into a single unit known as class. this concept promotes the bundling of data and methods that operate on that data within a single entity, ensuring that the internal state of object is protected and accessed only through defined interfaces.

Data hiding

Encapsulation facilitates data hiding by restricting direct access to object internal state from outside the class.

Private access specifiers ensure that sensitive data remains concealed, preventing unauthorized modification or access.

Access control

Encapsulation allows developers to define access levels for class members, determining their visibility and accessibility.

Access specifiers ensure controlled intraction with class attributes and methods, enhancing security and modularity.