OOPs - Object Oriented Programming System

Class - Blueprint

Ex : Blueprint of a Car

Object - Instance (Copy) of a Class

Ex : Car for Every Customer according to the Blueprint

Properties

Methods or Functions

4 Pillars

Encapsulation

Inheritance

Polymorphism

Abstraction

Access Specifiers

Public - Everywhere

Private - Within the Class \_

Protected - Within the Current Package and all other sub packages \_\_

If Java program is not mentioned as public class - then it is only accessible upto the current packages

Static - Using this a method can be called by using the classname.method without creating an Object

Constructor - It is used to instantiate the object and it is called by default when a object is created by default and it is a special type of method

this.varName - refers to current Instance

Encapsulation - Hides the internal/unwanted details using access specifiers i.e, bundling the data and methods into a single unit and can be read using getters and setters

Ex : Imagine changing the bank balance of a person by calling the balance variable of the class directly so inorder to protect that we use the method of Encapsulation

It enables everything by Methods

Inheritance -

Parent Constructor can be called by using super(varName)

Parent Method can be called by using super.methodName(parameters if any)

Base Class -> Sub Class

Reusing the methods , Properties

keyword : extends

Polymorphism -

Overriding the methods of the parent class according to the child class

Each child class can have its own implementation

Code Reusability

Extends the functionality

Compile Time Polymorphism / Method Overloading - multiple methods with the same name inside the same class having different parameters

Run Time Polymorphism / Method Overriding - implementing the method present in the parent class which is already defined in child class

Abstraction -

Hides complex implementation details and exposes only the essentials features

Shows what is necessary and keeps internal things hidden

It hides complexity

Simplifies Interaction

and it should be implemented by inheriting

Abstract Classes - Cannot be instantiated directly

Contains Abstract Methods(method without implementation) and regular methods (method with implementation)

It can be only used by sub Classes ensuring each subclass provides it implementation

Interface -

keyword - implements

It cotains only abstract methods

Multiple Inheritance

A static method can be called using class.method or Interface.method directly without creating an object