

# Introduction to Object Oriented Programming

Object-oriented programming (OOP) is a programming paradigm using "objects" - data structures consisting of data fields and methods together with their interactions - to design applications and computer programs. Programming techniques may include features such as data abstraction, encapsulation, messaging, modularity, polymorphism, and inheritance. Many modern programming languages now support O O P.

# Programming paradigm :

A programming paradigm is a fundamental style of computer programming. Paradigms differ in the concepts and abstractions used to represent the elements of a program (such as objects, functions, variables, constraints, etc.) and the steps that compose a computation (assignment, evaluation, continuations, data flows, etc.).

# Classes:

In object-oriented programming, a class is a construct that is used as a blueprint to create instances of the class (class instances, class objects, instance objects or just objects). A class defines constituent members which enable class instances to have state and behavior. Data field members (member variables or instance variables) enable a class object to maintain state. Other kinds of members, especially methods, enable a class object's behavior. Class instances are of the type of the associated class.

For example, an instance of the class "Fruit" (a "Fruit" object) would be of the type "Fruit". A class usually represents a noun, such as a person, place or (possibly quite abstract) thing. Programming languages that include classes as a programming construct subtly differ in their support for various class-related features. Most support various forms of class inheritance. Many languages also support advanced encapsulation control features, such as access specifiers.

# Object:

Object is an run time entity.

Is an Instance of class

Represents a Place ,Person ,anything that have some attributes.

# Objects and Instances:

There is a very important distinction between an object and an instance of an object. An object is actually a definition, or a template for instances of that object. An instance of an object is an actual thing that can be manipulated. For instance, we could define a Person object, which may include such member data as hair color, eye color, height, weight, etc. An instance of this object could be "Dave" and Dave has values for hair color, eye color, etc. This allows for multiple instances of an object to be created.


# Data Encapsulation and Abstraction:

Data encapsulation, sometimes referred to as **data hiding**.

Data Encapsulation and Data Abstraction is one of the most striking feature of object oriented programming.

The wrapping up of data and code into a single unit is called data encapsulation. The data is not accessible to the outside world only those functions which are wrapped into a class can only access the private data of the class.

Contd...



The concept of data encapsulation is supported in C++ through the use of the public, protected and private keywords which are placed in the declaration of the class.

**Note :**

- ❖ Anything in the class placed after the public keyword is accessible to all the users of the class
- ❖ Elements placed after the protected keyword are accessible only to the methods of the class or classes derived from that class
- ❖ Elements placed after the private keyword are accessible only to the methods of the class.

# Inheritance :


Inheritance is one of the most striking feature of object oriented programming.

Inheritance is the process by which one class can acquire the properties of another class.

The new classes, known as subclasses (or derived classes), inherit attributes and behavior of the pre-existing classes, which are referred to as superclasses (or ancestor classes). The inheritance relationships of classes gives rise to a hierarchy

Contd...





A **superclass**, base class, or parent class is a class from which other classes are derived. The classes that are derived from a superclass are known as **child classes**, **derived classes**, or **subclasses**.

In **object-oriented programming (OOP)**, inheritance is a way to compartmentalize and **reuse** code by creating collections of attributes and behaviors called objects which can be based on previously created objects.