1) Object :

Object is the basic unit of object-oriented programming. Objects are identified by its unique name. An object represents a particular instance of a class. There can be more than one instance of an object. Each instance of an object can hold its own relevant data.

An Object is a collection of data members and associated member functions also known as methods.

For example whenever a class name is created according to the class an object should be created without creating object can't able to use class.

The class of Dog defines all possible dogs by listing the characteristics and behaviors they can have; the object Lassie is one particular dog, with particular versions of the characteristics. A Dog has fur; Lassie has brown-and-white fur.

2) Class :

Classes are data types based on which objects are created. Objects with similar properties and methods are grouped together to form a Class. Thus a Class represents a set of individual objects. Characteristics of an object are represented in a class as Properties. The actions that can be performed by objects become functions of the class and is referred to as Methods.

For example consider we have a Class of Cars under which Santro Xing, Alto and WaganR represents individual Objects. In this context each Car Object will have its own, Model, Year of Manufacture, Colour, Top Speed, Engine Power etc., which form Properties of the Car class and the associated actions i.e., object functions like Start, Move, Stop form the Methods of Car Class.No memory is allocated when a class is created. Memory is allocated only when an object is created, i.e., when an instance of a class is created.

1.

3) Data abstraction & Encapsulation :

The wrapping up of data and its functions into a single unit is called Encapsulation.

When using **Data Encapsulation**, data is not accessed directly, it is only accessible through the functions present inside the class.

Data Abstraction increases the power of programming language by creating user defined data types. Data Abstraction also represents the needed information in the program without presenting the details.

Abstraction refers to the act of representing essential features without including the background details or explanation between them.

For example, a class Car would be made up of an Engine, Gearbox, Steering objects, and many more components. To build the Car class, one does not need to know how the different components work internally, but only how to interface with them, i.e., send messages to them, receive messages from them, and perhaps make the different objects composing the class interact with each other.

4) Inheritance :

Inheritance is the process of forming a new class from an existing class or base class.

The base class is also known as parent class or super class, the new class that is formed is called derived class.

Derived class is also known as a child class or sub class. Inheritance helps in reducing the overall code size of the program, which is an important concept in object-oriented programming.

It is classifieds into different types, they are

- **Single level inheritance**
- **Multi-level inheritance**
- **Hybrid inheritance**
- **Hierarchial inheritance**

5) Polymorphism :

Polymorphism allows routines to use variables of different types at different times. An operator or function can be given different meanings or functions. Polymorphism refers to a single function or multi-functioning operator performing in different ways.

Poly a Greek term ability to take more than one form. Overloading is one type of Polymorphism. It allows an object to have different meanings, depending on its context. When an exiting operator or function begins to operate on new data type, or class, it is understood to be overloaded.

Delegate : A delegate is a type that represents references to methods with a particular parameter list and return type. When you instantiate a delegate, you can associate its instance with any method with a compatible signature and return type. You can invoke (or call) the method through the delegate instance.

Delegates are used to pass methods as arguments to other methods. Event handlers are nothing more than methods that are invoked through delegates. You create a custom method, and a class such as a windows control can call your method when a certain event occurs. The following example shows a delegate declaration: