**Concepts Of OOPs for C#:**

**Inheritance** is a process in which one object acquires all the properties and behaviours of its parent object automatically. The class which inherits the members of another class called Derived Class and the class whose members are inherited called Base Class.

**Polymorphism** means multiple forms or methods with same names. It is useful for code reusability: reuse fields and methods of an existing class when you create a new class. It can be Static (Compile Time/Early Binding) or dynamic (Run time/Late Binding). The Compile Time techniques are 2 types. Function Overloading & Operator Overloading. Runtime technique is Method Overriding.

**Operator Overloading**: Creating 2 or more members having same name but different parameters called as member Overloading.

**Method Overloading**: Creating 2 or more methods with same name but different parameters called as Method Overloading.

**Method Override**: In this technique ‘override’ keyword will be used. By doing this process the derived class will override the base class.

**Encapsulation** is the concept of wrapping the data into a single unit. It collects data members and member functions into a single unit called class. The purpose of encapsulation is to prevent alteration of data from outside. This is implemented by using Access Specifiers.

**Public**: No restriction for access.

**Protected**: Accessible within in sub class in case of inheritance.

**Internal**: Accessible anywhere within the assembly.

**Private**: Restrictive and accessible only in class where it is declared.

**Protected Internal**: Accessible with in the assembly where it is declared also accessible with in derived class in another assembly.

Data **Abstraction** is the process of hiding certain details and showing only essential information to the user. Abstraction can be achieved with either **abstract classes** or **interfaces**.

**Abstract Class** declared with Abstract keyword. It can’t be instantiated (Creating Object). Its implementation must be provided by derived classes (inherited from another class). **Abstract method**: can only be used in an abstract class, and it does not have a body. The body is provided by the derived class (inherited from). An abstract class can have both abstract and regular methods:

**Interfaces** cannot be used to create objects. On implementation of an interface, you must override all its methods. Interface methods do not have a body - the body is provided by the "implement" class. Interface members are by default abstract and public. An interface cannot contain a constructor (as it cannot be used to create objects)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* xx \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Class** is nothing but an object at runtime which contains methods, constructors, fields etc.,

**Object** is an instance of a class. All members of the class can be accessed through the object.