OOP

Features of OOP

- Class
  - User Defined Type(UDT) with functions, fields, properties and events
    Fields will be private to the class.
    Methods and Properties will help in accessing and manipulating the fields of the class.
    Events are actions performed on the object.
  - U need an object to use the members of the class. The object is called as INSTANCE OF THE CLASS
  - Classes will have all the features of OOP Principles in it:
    Inheritance: The Class that is extended is called BASE CLASS/PARENT CLASS and the Class that provides the Extended Feature is called as DERIVED CLASS/CHILD CLASS/SUB-CLASS
- Access specifiers
  - Allows the members of the class to have accessibility defined at various levels.
    This allows some features of the class to hide and allow some features to be exposed. The hidden features are more like utilities that can be used within the class for its functionality.
    Exposed Features are are more like functions that can be consumed by the objects of the class.
  - 5 Access specifiers:
    public : Allows the members to be accessed thru' an object from any part of the app.
    private: Allows the members to be used only within the Class. Helps for modularity in the code(HELPER FUNCTIONS).
    protected: Allows the members of the class and its derived class to use them. Objects of the class created outside it cannot use them.
    internal: Similar to Friend functions, these members are available within the Assembly of the Project.
    protected internal: Mix of both internal and protected.
- Abstract classes
  - Classes with one or more fns with no implementation(Abstract methods). These classes are incomplete and cannot be used directly(They cannot be instantiated).
    U can extend this class into another and allow the implementation for those abstract methods in it. This extended class would be the concrete Implementation of the method.
- Interfaces
  - Interfaces are abstract classes that have only abstract methods in them.
    Interfaces are created to allow classes to provide concrete implementations of those methods. In this case, it is like a contract where the class that implements the interface must/should provide the public implementation of all the methods of that interface.
    All Interfaces are abstract classes, the reverse is not the same
- Method Overloading & Method Overriding
  - If a Class has multiple methods with similar functionalities with difference in the inputs given to the function to perform its operation, then U can create functions with a same name/general name and give different arguments to that function.
    This is called as METHOD OVERLOADING.
  - As a feature of Inheritance, a method can be modified by the derived class thru' a concept called METHOD OVERRIDING.

Problems of Functional Programming

- Reusability
- Extendability
- Simplicity
- Maintainability

SOLID Principles

Single Responsibility	Layered architecture and modularity
Open-Closed	Inheritance Feature of OOP
Luskov's Substitution	Polymorphism
Interface Segregation	Dividing UR Functions into small interfaces.
Dependency Inversion	Abstraction/Interface objects in UR Class

Inheritance points
1. .NET does not support Multiple Inheritance => A Class can have only one base class at any level.
2. .NET supports Multi-level Inheritance. => U can have a chain of hierarchy in UR class derivation.
3. There is no public or private inheritance in .NET. All the members of the base class retain their accessibility in the derived classes.