# Type of Class

## 1. Regular Classes:

Definition: Regular classes are fundamental building blocks in C# that encapsulate data and behavior into a single unit.

Use: They are used to model entities, concepts, or components in a software system.

### Features:

Encapsulation of data and behavior.

Inheritance: They can be used as base classes for other classes.

Polymorphism: They can participate in method overriding and method overloading.

Access Modifiers: Control the visibility of class members.

Constructors and Destructors: Initialize and clean up class instances.

### Advantages:

Well-suited for general-purpose programming.

Provide a clear structure for organizing code.

Supports inheritance and polymorphism for code reuse and extensibility.

Real-Life Example:

Consider a Car class in an automobile management system. It encapsulates attributes like make, model, year, and behaviors like startEngine(), accelerate(), and brake().

## 2. Abstract Classes:

Definition: Abstract classes are classes that cannot be instantiated directly but provide a blueprint for other classes. They may contain abstract methods that must be implemented by derived classes.

Use: They are used to define common behavior and enforce a contract for derived classes.

### Features:

Can have abstract methods that are meant to be implemented by derived classes.

Can also have normal methods with a default implementation.

Cannot be instantiated directly but can be used as base classes.(Unable to create object)

### Advantages:

Encourage code reusability and enforce a consistent structure in derived classes.

Provide a way to define common behavior without full implementation.

Support inheritance and polymorphism.  
Disadvantages:

May lead to a deep inheritance hierarchy, which can be difficult to understand.

Limited to single inheritance, as a class can only inherit from one abstract class.

### Real-Life Example:

Consider an Animal class in a zoo management system. It could be an abstract class defining common attributes and methods for all animals, such as eat(), sleep(), and makeSound(). Subclasses like Lion, Elephant, and Monkey would implement their specific behavior.

## 3. Static Classes:

Definition: Static classes are classes that cannot be instantiated and contain only static members.

Use: They are used to group related utility methods or constants.

### Features:

Cannot be instantiated.

Contain only static members: fields, methods, properties.

Cannot contain instance constructors.

### Advantages:

Provide a way to organize utility functions without the need for object instantiation.

Help in improving performance as they are loaded into memory only once.

Prevent accidental instantiation and enforce a singleton-like behavior.

### Disadvantages:

Cannot be used to maintain state across multiple calls (since they cannot have instance fields).

Limited flexibility compared to regular classes.

### Real-Life Example:

Consider a MathUtils class containing static methods like Add(), Subtract(), Multiply(), and Divide(). These methods perform mathematical operations without needing to instantiate an object.

## 4. Partial Classes:

Definition: Partial classes allow the splitting of a class's definition into multiple files.

Use: They are used to separate auto-generated code from user-written code or to organize large classes.

### Features:

The class definition can be divided into multiple files.

All partial declarations must be marked with the partial keyword.

All parts are combined at compile time into a single class definition.

### Advantages:

Useful in large projects where multiple developers can work on different parts of a class.

Allow separation of generated code and manually written code.

### Disadvantages:

Overuse of partial classes can make code harder to understand.

May lead to fragmentation and difficulty in locating related code.

### Real-Life Example:

Consider a User class in a web application. One partial class file could contain properties and methods related to authentication, while another partial class file could contain properties and methods related to profile management.

## 5. Sealed Classes:

Definition: Sealed classes are classes that cannot be inherited.

Use: They are used when you want to prevent further derivation of a class.

### Features:

Cannot be used as a base class for other classes.

### Advantages:

Prevents unintended derivation and modification of class behavior.

Guarantees the integrity of the class's implementation.

### Disadvantages:

Limits extensibility and flexibility compared to regular classes.

### Real-Life Example:

Consider a Rectangle class representing geometric shapes. If you want to ensure that no other class can inherit from Rectangle, you can declare it as sealed, ensuring that its behavior remains consistent.