Static and Private Constructor

Arctech Info Private Limited

Static Constructor

- Static constructors are used to initialize the static members of the class and are implicitly called before the creation of the first instance of the class. Non-static constructors are used to initialize the non-static members of the class.
- **Declaration**: Static constructors are declared using a static modifier explicitly while all other remaining constructors are non-static constructors. Non-static constructors can also be called as Instance Constructors as they need instance to get executed.

Static Constructor

- **Calling**: Static constructors are always called implicitly but the non-static constructors are called explicitly i.e by creating the instance of the class.
- **Execution**: Static constructor executes as soon as the execution of a class starts and it is the first block of code which runs under a class. But the non-static constructors executes only after the creation of the instance of the class. Each and every time the instance of the class is created, it will call the non-static constructor.

Static Constructor

- **Times of Execution**: A static constructor will always execute once in the entire life cycle of a class. But a non-static constructor can execute zero time if no instance of the class is created and n times if the n instances are created.
- **Parameters**: We cannot pass any parameters to the static constructors because these are called implicitly and for passing parameters, we have to call it explicitly which is not possible
- **Initialization of fields**: Static constructors are used to initialize the static fields and non-static constructors are used to initialize the non-static fields.
- **Overloading**: Non-static constructors can be overloaded but not the static constructors. Overloading is done on the parameters criteria. So if you cannot pass the parameters to the Static constructors then we can't overload it.

Private Constructor

- A constructor in C# is a special method of a class that is used to instantiate a class. A class constructor, just like other class methods, can be public, private, or protected.

- Uses of Private Constructor
  - When all the members of a class are static
  - When you don't want a class to be inherited by other classes
  - When you want to implement singleton design pattern

- **1. When All the Class Members are Static**
- A member of a class which can be a method or a variable can be static or non-static. A static member of a class can be accessed using the class name. You don't need to instantiate a class in order to access its static member. Hence, if all the members of a class are static and can be accessed via the class name, you can add a private constructor to prevent class instantiation.

- **2. When You Don't want a Class to Be Inherited**
- Adding a private constructor prevents a class from being inherited by other classes

3. While Implementing a Single Design Pattern

- You can use a private constructor when you want to implement a single design pattern. In a single design pattern, only one object of a particular class is created.