Constructor in C#

What is Constructor?

A constructor is a special method of the class which gets automatically invoked whenever an instance of the class is created. It is used to assign initial values to the **data members** of the same class.

Important points to Remember About Constructors

- Constructor of a class must have the same name as the class name in which it resides.
- A constructor cannot be abstract, final, and Synchronized.
- Within a class, you can create only one static constructor.
- A constructor doesn't have any return type, not even void.
- A static constructor cannot be a parameterized constructor.
- A class can have any number of constructors.
- Access modifiers can be used in constructor declaration to control its access i.e which other class can call the constructor.

Declaration

<Access Modifer> ClassName(parameters)

{

}

Types of Constructors

1. **Default Constructor –** No parameters initial value will be 0 for int null for string
2. **Parameterized Constructor –** At least one param
3. **Copy Constructor -** This constructor creates an object by copying variables from another object. Its main use is to initialize a new instance to the values of an existing instance.
4. **Private Constructor -** If a constructor is created with private specifier is known as Private Constructor. It is not possible for other classes to derive from this class and it's not possible to create an instance of this class.
   - Implementation of Singleton class pattern
   - use private constructor when we have only static members.
   - Using private constructor, prevents the creation of the instances of that class.

5. **Static Constructor -** Static Constructor must be invoked only once in the class and it has been invoked during the creation of the first reference to a static member in the

class. A **static constructor** is initialized static fields or data of the class and to be executed only once.

- It can't be called directly.
- When it is executing then the user has no control.
- It does not take access modifiers or any parameters.
- It is called automatically to initialize the class before the first instance created.

Example:

```
using System;

namespace C_Basics
{
    class User
    {
        public string Name{ get; set; }
        public int Id { get; set; }

        public static int Sequence { get; set; }

        public User(string name, int id) // Parameterized constructor.
        {
            this.Name = name;
            this.Id = id;
        }

        public User(User user) // Copy constructor
        {
            Name = user.Name;
            Id = user.Id;
        }

        private User()
        {
        }

        static User()
        {
            Console.WriteLine("Static Constructor");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            User user = new User("Athulya", 101);
            User userCopy = new User(user);
            //User userPrivate = new User("athulya"); -- Not Possible due to 
the Private
            User.Sequence = 1000; // Classname.Static Variable: Not possible 
to call a static variable with object.
            User.Sequence = 5000;
            Console.WriteLine("Param Constructor Data!\nName: " +user.Name + 
"\nId: " + user.Id );
            Console.WriteLine("Copy Constructor Data!\nName: " + userCopy.Name 
+ "\nId: " + userCopy.Id);
            Console.WriteLine("Private Constructor -Static Data access only!\n 
Sequence: " + User.Sequence);
        }
    }
}
```

Output

```
Static Constructor

Param Constructor Data!

Name: Athulya

Id: 101

Copy Constructor Data!

Name: Athulya

Id: 101

Private Constructor -Static Data access only!

 Sequence: 5000
```