**Static Classes**  
A static class can be created using a keyword called "static" used at the class definition. A static class can contain only static members (static data members and static methods). You can‘t create an object for the static class.

**Advantages**

1. If you declare any non-static member, it generates a compile time error, so that it is guaranteed that the class contains only static members; there is no chance of declaring a non-static member accidentally.
2. When you try to create an instance to the static class, it again generates a compile time error, because the static members can be accessed directly with its class name.

namespace ExampleOfStaticClass {

// Creating static class

// Using static keyword

static class Author {

    // Static data members of Author

    public static string A\_name = "Bibek";

    public static string L\_name = "CSharp";

    public static int T\_no = 84;

    // Static method of Author

    public static void details()

    {

        Console.WriteLine("The details of Author is:");

    }

}

// Driver Class

public class GFG {

    // Main Method

    static public void Main()

    {

        // Calling static method of Author

        Author.details();

        // Accessing the static data members of Author

        Console.WriteLine("Author name : {0} ", Author.A\_name);

        Console.WriteLine("Language : {0} ", Author.L\_name);

        Console.WriteLine("Total number of articles : {0} ",

                                              Author.T\_no);

    }

}

public static class MyStaticClass

{

public static int myStaticVariable = 0;

public static void MyStaticMethod()

{

Console.WriteLine("This is a static method.");

}

public static int MyStaticProperty { get; set; }

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine(MyStaticClass.myStaticVariable);

MyStaticClass.MyStaticMethod();

MyStaticClass.MyStaticProperty = 100;

Console.WriteLine(MyStaticClass.MyStaticProperty);

}

}

Static members in non-static class

public class MyNonStaticClass

{

private static int myStaticVariable = 0;

public static void MyStaticMethod()

{

Console.WriteLine("This is static method.");

}

public void myNonStaticMethod()

{

Console.WriteLine("Non-static method");

}

}

C# Static Constructor

A static or non-static class can have a static constructor without any access modifiers like public, private, protected, etc.

A static constructor in a non-static class runs only once when the class is instantiated for the first time.

A static constructor in a static class runs only once when any of its static members accessed for the first time.

public static class MyStaticClass

{

static MyStaticClass()

{

Console.WriteLine("Inside static constructor.");

}

public static int myStaticVariable = 0;

public static void myStaticMethod()

{

Console.WriteLine("This is static method.");

}

public static int MyStaticProperty { get; set; }

}

class Program

{

static void Main(string[] args)

{

MyStaticClass.myStaticVariable = 100;

MyStaticClass.MyStaticProperty = 200;

MyStaticClass.myStaticVariable = 300;

MyStaticClass.MyStaticProperty = 400;

}

}

1. Static classes cannot be instantiated using the new keyword
2. Static items can only access other static items. For example, a static class can only contain static members, e.g., variables, methods, etc. A static method can only contain static variables and can only access other static items.
3. Static items share the resources between multiple users.
4. Static cannot be used with indexers, destructors or types other than classes.
5. A static constructor in a non-static class runs only once when the class is instantiated for the first time.
6. A static constructor in a static class runs only once when any of its static members accessed for the first time.
7. Static members are allocated in high frequency heap area of the memory.