When To Use Static Classes In C#

C# supports static classes and static members. A static class can't be instantiated. A C# class can have static or non-static members. A static member has only one copy of the member, regardless of the number of instances. Static members and their values belong to the type itself, rather than the object. If multiple instances of a class are created, the last updated value of a static member will be available to all instances.

The static modifier in C# declares a static member of a class. The static modifier can be used with classes, properties, methods, fields, operators, events, and constructors, but it cannot be used with indexers, finalizers, or types other than classes.

The static keyword in C# language is used to declare a static class. Learn more about static keyword here Static Keyword In C#.

Static Class

A static class cannot be instantiated. All members of a static class are static and are accessed via the class name directly, without creating an instance of the class.

The following code is an example of a static class, CSharpCorner. We know that all members of the class are static.

```
01. public static class CSharpCorner
02. {
03. // Static fields
04. public static string Name = "C# Corner";
05. public static string Founder = "Mahesh Chand";
06. public static DateTime YearFounded = new DateTime(2000, 01, 01);
07. public static string Location = "Downingtown, PA";
08. public static string Description =
09. "Online community of software and data developers";
10. public static int GetAgeOfWebsite()
11. {
12. return 1;
13. }
14. }
```

Static classes have the following characteristics:

- Static classes cannot contain Instance Constructors.
- Static classes contain only static members.

- Static classes cannot be instantiated.
- Static classes are sealed. That means, you cannot inherit other classes from instance classes.

You can learn more about static classes here: Static Class in C#

Static Members

A static or non-static class static constructors, properties, methods, fields, operators, and events. Static properties and static methods are the most-used static members.

Static Constructor

Static constructor can't be parameterized.
Static constructor doesn't have any access modifier because it doesn't have message passing and is used during domain processing.
Static Constructor is used to initialize static data members of the class.

Static Field

A field with the static keyword represents a static field. Static fields can be declared as follows by using the static keyword.

```
01. public static class HistoryTeacher
02. {
03. // static field
04. public static string Subject = "History";
05. }
```

The following code calls a static field.

```
01. Console.WriteLine(HistoryTeacher.Subject);
```

When we declare static data members inside a class, it can be initialized with a value as shown above. All un-initialized static fields automatically get initialized to their default values when the class is loaded for the first time.

Static Property

Static properties are used to get or set the value of static data members of a class. The following code example declares four static properties.

```
01. public static class HistoryTeacher
02. {
03. // private fields
04. private static string name;
```

```
05.  private static string school;
06.  private static int rank;
07.  private static int years;
08.  // static properties
09.  public static int Years { get => years; set => years = value; }
10.  public static int Rank { get => rank; set => rank = value; }
11.  public static string School { get => school; set => school = value; }
12.  public static string Name { get => name; set => name = value; }
13.  }
```

The following code example sets static property values.

```
01.  HistoryTeacher.Name = "Mahesh Chand";
02.  HistoryTeacher.Rank = 2;
03.  HistoryTeacher.School = "Garnet Valley High School";
04.  HistoryTeacher.Years = 5;
```

In the same way, you can access a static property by using the class name.

```
01.  Console.WriteLine(HistoryTeacher.Name);
```

Static Method

Static methods are shared methods. They can be called with the class name and static method name only. You cannot instantiate a static method.

Static methods only use static data members to perform calculation or processing.

The following code example declares a static method that takes two int values as method arguments.

```
01.  public static class HistoryTeacher
02.  {
03.  // static method
04.  public static int CalculateScore(int rank, int years)
05.  {
06.  return rank * years;
07.  }
08.  }
```

The following code example calls a static method.

```
01.  Console.WriteLine(HistoryTeacher.CalculateScore(3, 5));
```

Why use static classes and static members?

Static classes are used as containers for static members. Static methods and static properties are the most-used members of a static class. All static members are called directly using the

class name. Static methods do a specific job and are called directly using a type name, rather than the instance of a type.

Here is a list of few use cases of static classes.

- A Math class with all static methods. Static classes are useful and provide an easy way to access its members that does not need to work differently for different objects.
- Above listed CSharpCorner class. We know the value of CSharpCorner class members such as its founder, launch date, location, and description, will never change regardless of its objects.
- App Configuration class that has all static settings about an app and the values of settings don't change based on the objects or users.
- A DatabaseConfig class that may have members such as database name, server name, port number, and even a connection string. We know that these values will not change for objects.

Static classes and static members are useful because they do not require instances created for each new object. That means, they consume fewer resources and no duplication of the same class or member is needed in memory.

Static members make code cleaner.

In theory, a static should give better performance compare to an instance. However, it is unnoticeable.

Complete Code Example

Here is a complete program that shows how to use static class and static members.

```
01. using System;
02. namespace StaticInCSharp
03. {
04. class Program
05. {
06. static void Main(string[] args)
07. {
08. Console.WriteLine(HistoryTeacher.Subject);
09. HistoryTeacher.Name = "Mahesh Chand";
10. HistoryTeacher.Rank = 2;
11. HistoryTeacher.School = "Garnet Valley High School";
12. HistoryTeacher.Years = 5;
13. Console.WriteLine(HistoryTeacher.Name);
14. Console.WriteLine(HistoryTeacher.CalculateScore(3, 5));
15. Console.ReadKey();
16. }
17. }
18. public static class HistoryTeacher
19. {
```

```
20.  // static field
21.  public static string Subject = "History";
22.  // private fields
23.  private static string name;
24.  private static string school;
25.  private static int rank;
26.  private static int years;
27.  // static properties
28.  public static int Years { get => years; set => years = value; }
29.  public static int Rank { get => rank; set => rank = value; }
30.  public static string School { get => school; set => school = value; }
31.  public static string Name { get => name; set => name = value; }
32.  // static method
33.  public static int CalculateScore(int rank, int years)
34.  {
35.  return rank * years;
36.  }
37.  }
38.  }
```

Summary

In this article, I discussed what static classes are in C# and how to use a static class in your C# application.

Next Recommended Article

Static Class In C#

static class | static keyword | static method | static modifier

Mahesh Chand Admin

Founder: C# Corner, Mindcracker

https://www.c-sharpcorner.com

167.9m 1 MVP 13

26 8

Type your comment here and press Enter Key (Minimum 10 characters)

Very informative...want to add one point: we should avoid static variables for configuration or db

connections (user specific) when it is shared with multi user. i.e public static userName {get; set;} so, multiple user are accessing it, every time it will override userName variable. Users will not get correct userName.

Jayesh Agrawal

• 543 • 3.3k • 337.5k

May 13, 2019

0 1 Reply

Yes that's correct.

Mahesh Chand

• Admin • 385.6k • 167.9m

May 13, 2019

0

I'd be a little concerned about using static classes for configuration or database connections. The only use for those classes is for other classes to depend on them. Other classes that depend on static classes can be very difficult to test. Static classes are good for static methods, as you mentioned, like math.

Scott Hannen

• 1900 • 10 • 0

May 03, 2019

1 1 Reply

How and what problems do you see in that case? I've used a Static class in a DLL that has some constant database and other settings that are constant values. All caller programs use that class to access the properties.

Mahesh Chand

• Admin • 385.6k • 167.9m

May 03, 2019

0

Thank you. very useful information. I think there is on typo in statement "A Match class with all static methods", here expected "Math" in place of "Match".

Gajanan Chavhan

• 1199 • 773 • 43.1k

May 02, 2019

0 1 Reply

Thank you, Gaganan. Appreciate it.

Mahesh Chand

• Admin • 385.6k • 167.9m

May 02, 2019

1

It would be interesting to look deeper at the decision making process for deciding whether or not to use a static class, taking into consideration that you do not want to create a static class only to change it to a non-static class later.

Geoffrey Payne

• 1905 • 5 • 0

May 02, 2019

1 1 Reply

Thanks. Good point. Use of static classes is not very common on day to day applications. I've hardly used them. May be some static members for settings etc. Components and library developers may use them for certain classes. For example, Microsoft has used them in their .NET libraries.

Mahesh Chand

• Admin • 385.6k • 167.9m

May 02, 2019

0