<CodingSonata />

6 Types of Constructors in C#

Aram Tchekrekjian

@AramT87

1

Default Constructor

@AramT87

Default Constructor

It is commonly known as the parameterless constructor or the instance constructor.

This constructor is called when creating a new instance of the class.

A default constructor would initialize the fields to their default values:

-> Zero for numeric types
-> null for reference types

@AramT87

Default Constructor

You can override the default behavior by including your own initialization code within the default constructor

```
public QuestionModel()
{
    Console.WriteLine
("Inside the default constructor");

}
```

@AramT87

2

Parametrized Constructor

@AramT87

Parametrized Constructor

This is similar to the default constructor, and shares one of the names - instance constructor.

The difference is that it takes input from outside the class

Also you can define multiple overloads of the parametrized constructor

@AramT87

Parametrized Constructor

```
public QuestionModel(int id,
                     string title,
                     string description)
{
    Id = id;
    Title = title;
    Description = description;
    IsActive = true;
    PostingDate = DateTime.Now;
}
```

@AramT87

3

Private Constructor

@AramT87

Private Constructor

It is a special constructor marked as private, so it is inaccessible from outside the class

If the class doesn’t have a parametrized constructor, creating a new instance from the class with private constructor throws an exception.

@AramT87

Private Constructor

A private constructor can be used only with static members within the class

Some popular implementations for private constructors are through Singletons and Factories

A private constructor restricts a class from being inherited

@AramT87

Private Constructor

```
public class SimpleSingleton
{
    private static SimpleSingleton instance;
    private SimpleSingleton() { }

    public static SimpleSingleton GetInstance()
    {
        if (instance == null)
        {
            instance = new SimpleSingleton();
        }
        return instance;
    }
    public void DoSomething()
    {
        Console.WriteLine("Inside Singleton Method");
    }
}

// Invocation
SimpleSingleton.GetInstance().DoSomething();
```

4

Static Constructor

@AramT87

Static Constructor

Used to initialize static members of a class/struct

A static constructor is called before the instance constructor

This type of constructor is automatically called before the creation of the instance

Or when any static member is referenced

@AramT87

Static Constructor

The static constructor is called at most once even if multiple instances of the same class are created

This is mainly used to define a global value for a class and use that value in any subsequent instance

Can be useful to define some configurations for the type

@AramT87

Static Constructor

You cannot overload static constructor

You cannot inherit a static constructor

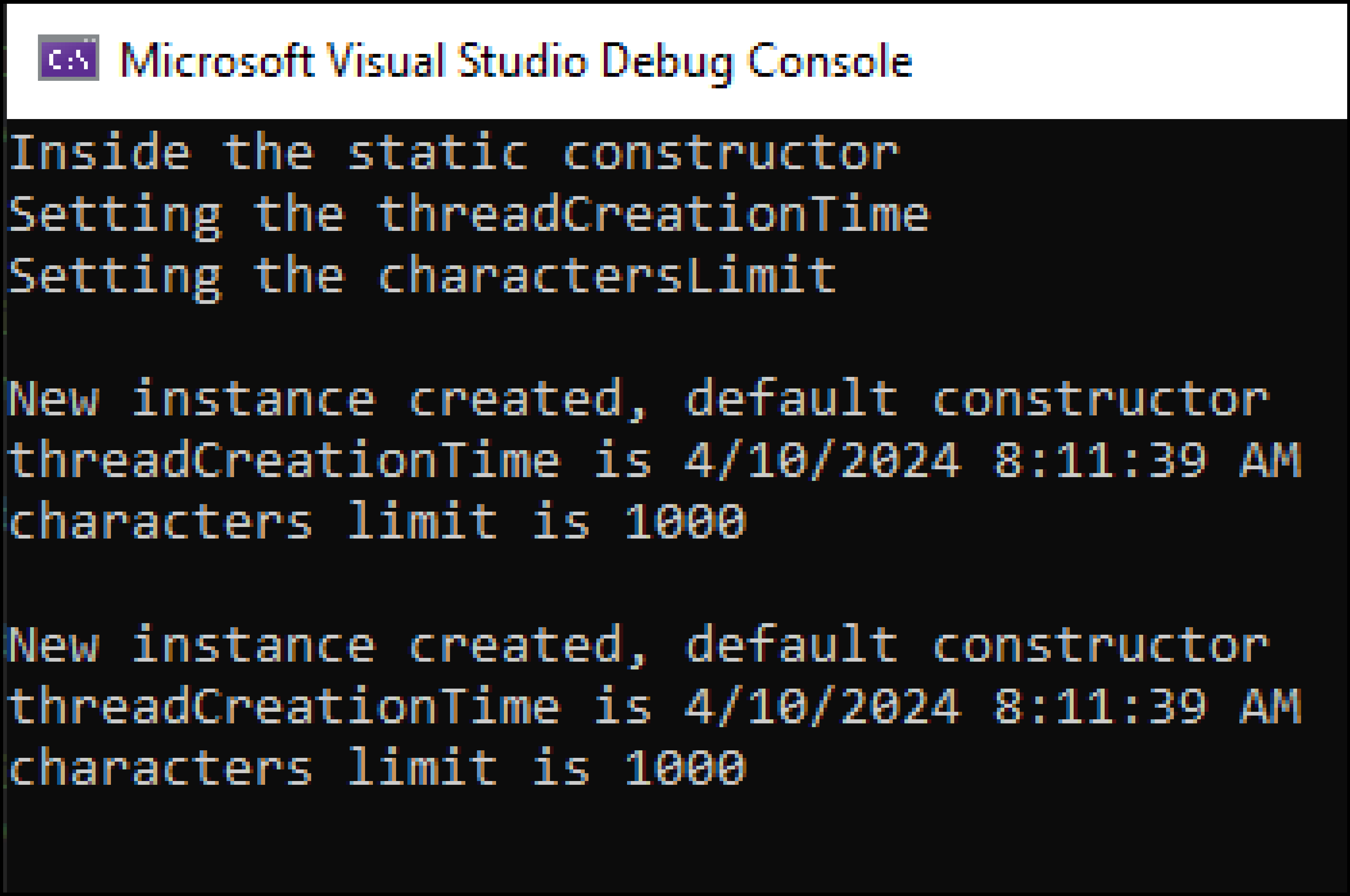
You cannot access a non-static field within a static constructor

Static Constructor

```csharp
public class QuestionModel
{
    static DateTime threadCreationTime;
    static short charactersLimit;

    static QuestionModel()
    {
        threadCreationTime = DateTime.Now;
        charactersLimit = 1000;

        Console.WriteLine("Inside the static constructor");
        Console.WriteLine($"Setting the threadCreationTime");
        Console.WriteLine($"Setting the charactersLimit \n");
    }
    public QuestionModel()
    {
        Console.WriteLine("New instance created, default constructor");
        Console.WriteLine($"threadCreationTime is {threadCreationTime}");
        Console.WriteLine($"characters limit is {charactersLimit} \n");
    }
}

// Invocation
QuestionModel? question1 = new();
QuestionModel? question2 = new();
```

@AramT87

Static Constructor


Microsoft Visual Studio Debug Console

```
Inside the static constructor
Setting the threadCreationTime
Setting the charactersLimit

New instance created, default constructor
threadCreationTime is 4/10/2024 8:11:39 AM
characters limit is 1000

New instance created, default constructor
threadCreationTime is 4/10/2024 8:11:39 AM
characters limit is 1000
```

@AramT87

5

Copy Constructor

@AramT87

Copy Constructor

This type of constructor was first introduced in C++

It is used to copy class members’ values to another class as to create a duplicate or clone

Copying data from instance to instance requires you decide the depth level of copy in your copy constructor - Shallow and Deep

@AramT87

Copy Constructor

Shallow copy copies values of the primitive data types and copies references of the reference types

Deep copy copies each of the primitive data types and creates new instance of the reference types and includes any nested reference types recursively

@AramT87

Copy Constructor

A copy constructor should include an instance of the class as parameter

And then inside it, you can manually assign each value from the input parameter instance to the matching member within the class

@AramT87

Copy Constructor

```csharp
public class QuestionModel
{
    public QuestionModel(QuestionModel questionModel)
    {
        Id = questionModel.Id;
        Title = questionModel.Title;
        Description = questionModel.Description;
        IsActive = questionModel.IsActive;
        PostingDate = questionModel.PostingDate;
    }

   // Code remove for brevity
}

QuestionModel? question1 = new(
    1,
    "How many words in English?",
    "This is the question, " +
    "how many words are there " +
    "in the English language?");

// Invocation
QuestionModel? question2 = new(question1);
```

@AramT87

6

Primary Constructor

@AramT87

Primary Constructor

This was introduced in C# 12

A primary constructor is a concise way to write parametrized constructor

This constructor's parameters have class or struct level scope

@AramT87

Primary Constructor

If the primary constructor’s parameters are accessed in a method within the class/struct, the compiler creates hidden fields to represent each parameter

Primary Constructor

You can use them to do different operations:

- Initialize Properties or Fields
- Initialize base class
- Specify Parameters in Dependency Injection

  (Useful with ASP.NET Core Controllers when you are injecting dependencies in controllers or services)

@AramT87

Primary Constructor

```
public class QuestionModel
    (int id,
    string? title,
    string? description,
    bool isActive,
    DateTime? postingDate)
{
    public int Id => id;
    public string? Title => title;
    public string? Description => description;
    public bool IsActive => isActive;
    public DateTime? PostingDate => postingDate;
}
```

@AramT87

Found this useful?

Consider Reposting

Thank You

Follow me for more content

Aram Tchekrekjian

Microsoft® MVP Most Valuable Professional

in AramT87

Get Free tips and Tutorials in .NET and C#

Join 600+ Readers

CodingSonata.com/newsletters

<CodingSonata />