C# Partial Method

In c#, the **partial method** is a special type of method that will contain a declaration part in one partial class (/tutorial/csharp/csharp-partial-class) and the definition part in another partial class (/tutorial/csharp/csharp-partial-class) or in the same partial class (/tutorial/csharp/csharp-partial-class).

Generally, the partial methods will exist either in partial class (/tutorial/csharp/csharp-classes-and-objects-with-examples) or struct (/tutorial/csharp/csharp-structures-structs), and they will contain two main parts one is **declaration** and **definition**. Here, the **declaration** part represents a signature of the partial method, and the **definition** part represents an implementation of the partial method.

In the partial method, the definition part is optional, so it may or may not have an implementation in partial classes (/tutorial/csharp/csharp-partial-class). In case the implementation part of the partial method is not available in any partial class. The compiler will remove the method's definition and all the calls related to that particular method in the final class.

Following is the example of implementing a partial method using `partial` keyword in two class (/tutorial/csharp/csharp-classes-and-objects-with-examples) files, **User1.cs** and **User2.cs**.

User1.cs

```
public partial class User
{
    // Partial Method Declaration
    partial void GetUserDetails(string a, string b);
    public void TestMethod()
    {
       Console.WriteLine("Test");
    }
}
```

If you observe the above code, we declared a partial method called **GetUserDetails** in **User1.cs** class file using `partial` keyword along with a standard method called **TestMethod()**.

User2.cs

```
public partial class User
{
     public User(string a, string b)
     {
        GetUserDetails(a, b);
    }
    // Partial Method Implementation
    partial void GetUserDetails(string x, string y)
    {
       Console.WriteLine("Name: " + x);
       Console.WriteLine("Location: " + y);
    }
}
```

If you observe the above code, we implemented a partial method called **GetUserDetails** in **User1.cs** class file using `partial` keyword (/tutorial/csharp/csharp-keywords-reserved-contextual) with required variables (/tutorial/csharp/csharp-variables-with-examples) and constructor (/tutorial/csharp/csharp-constructors-with-examples).

When you execute the above code, the compiler will combine these two partial classes into one **User** class as shown below.

User

```
public class User
{
    public User(string a, string b)
    {
        GetUserDetails(a, b);
    }
    public void TestMethod()
    {
        Console.WriteLine("Test");
    }
    private void GetUserDetails(string x, string y)
    {
        Console.WriteLine("Name: " + x);
        Console.WriteLine("Location: " + y);
    }
}
```

This is how the compiler will combine all the partial classes into a single class while executing the application in the c# programming language.

Rules to Implement Partial Method

In c#, we need to follow certain rules to implement the partial method in our applications.

- In c#, the partial method can be implemented within a partial class (/tutorial/csharp/csharp-classes-and-objects-with-examples) or structure (/tutorial/csharp/csharp-structures-structs) only, and the signatures in both parts of the partial type must be the same.
- The partial method declaration must begin with `partial` keyword.
- By default, partial methods are implicitly **private,** and they cannot be **virtual**.
- Partial methods can have ref (/tutorial/csharp/csharp-pass-by-reference-ref-with-examples) but not out (/tutorial/csharp/csharp-out-parameter-with-examples) parameters.

- Partial methods can be static (/tutorial/csharp/csharp-static-keyword) and **generic**.

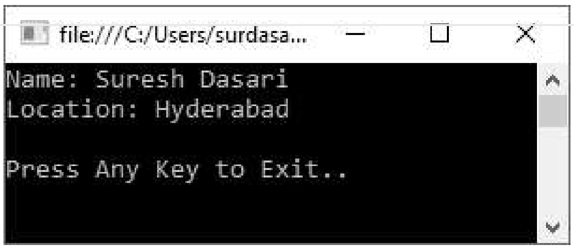
C# Partial Method Example

Following is the example of defining a partial method in a partial class (/tutorial/csharp/csharp-partial-class) using `partial` modifier in c# programming language.

```
using System;

namespace Tutlane
{
    public partial class User
    {
        // Partial Method Declaration
        partial void GetUserDetails(string a, string b);
    }

    public partial class User
    {
        public User(string a, string b)
        {
            GetUserDetails(a, b);
        }
        // Partial Method Implementation
        partial void GetUserDetails(string x, string y)
        {
           Console.WriteLine("Name: " + x);
           Console.WriteLine("Location: " + y);
        }
    }

    class Program
    {
       static void Main(string[] args)
       {
           User u = new User("Suresh Dasari", "Hyderabad");
           Console.WriteLine("\nPress Enter Key to Exit..");
           Console.ReadLine();
       }
    }
}
```

If you observe the above example, we created a partial method **GetUserDetails** in two partial classes (/tutorial/csharp/csharp-partial-class) using `partial` keyword to perform required operations.

When you execute the above c# program, we will get the result below.



```
Name: Suresh Dasari
Location: Hyderabad

Press Any Key to Exit..
```

This is how we can use partial methods in partial class (/tutorial/csharp/csharp-classes-and-objects-with-examples) or structure (/tutorial/csharp/csharp-structures-structs) when we have a method declaration and implementation in multiple files with `partial` modifier based on our requirements.

CONTACT US

Address: No.1-93, Pochamma Colony, Manikonda, Hyderabad, Telangana - 500089

Email: support@tutlane.com (mailto:support@tutlane.com)