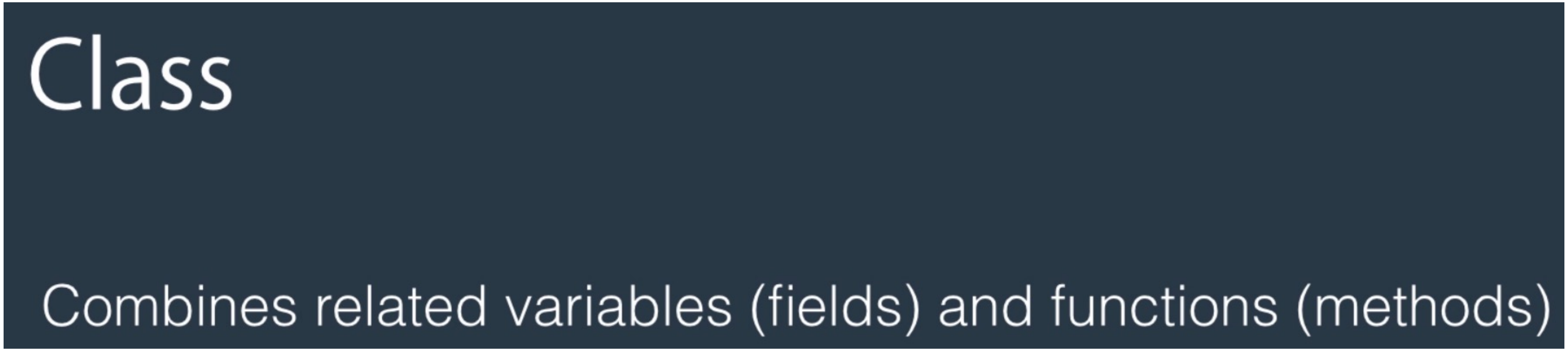
4. Non-Primitive Types
1. Classes

Non-Primitive Types

- Classes
- Arrays
- Structures
- Strings
- Enums
- Reference types versus Value Types
- Memory management of different types.

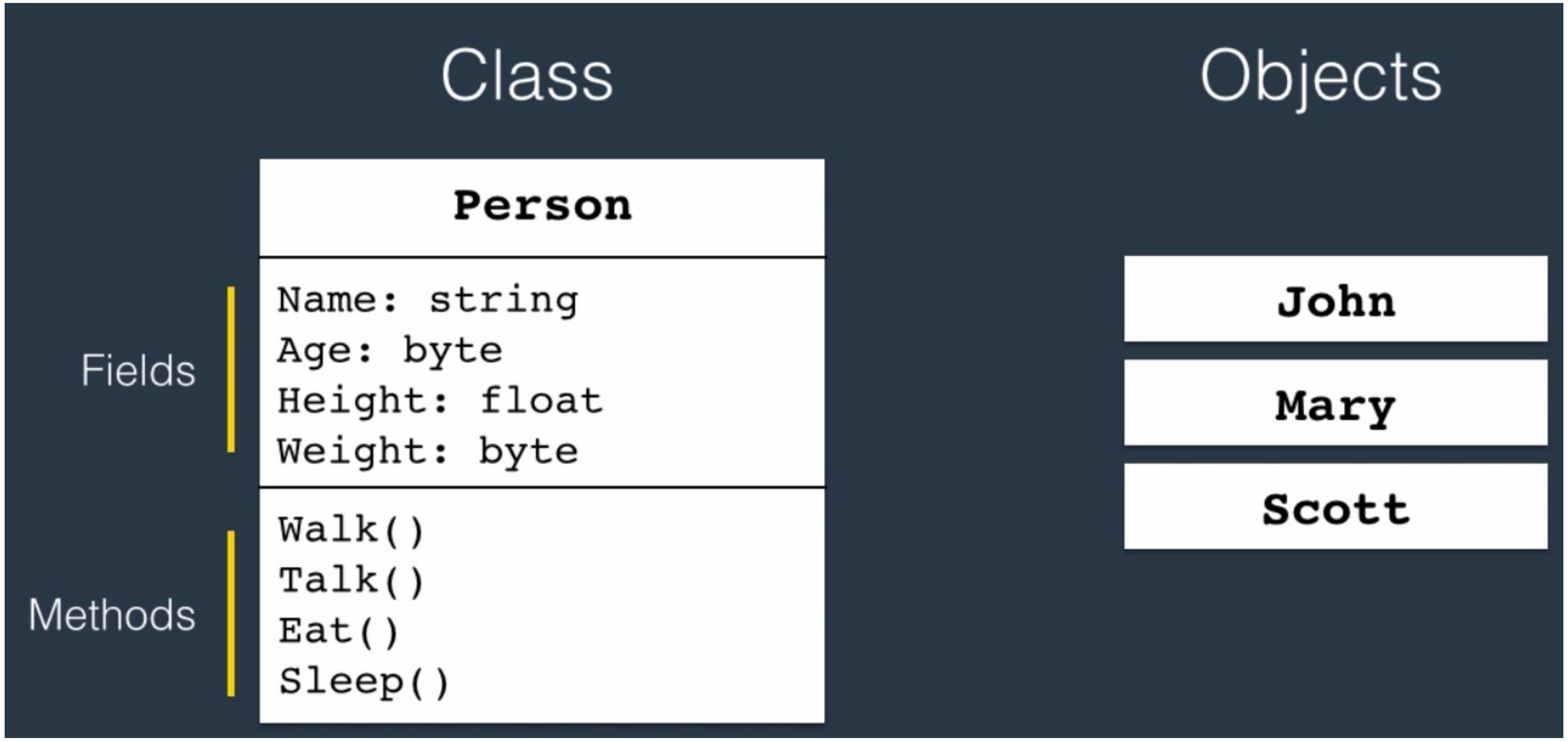
Classes - Classes are building blocks of our applications



Class

Combines related variables (fields) and functions (methods)

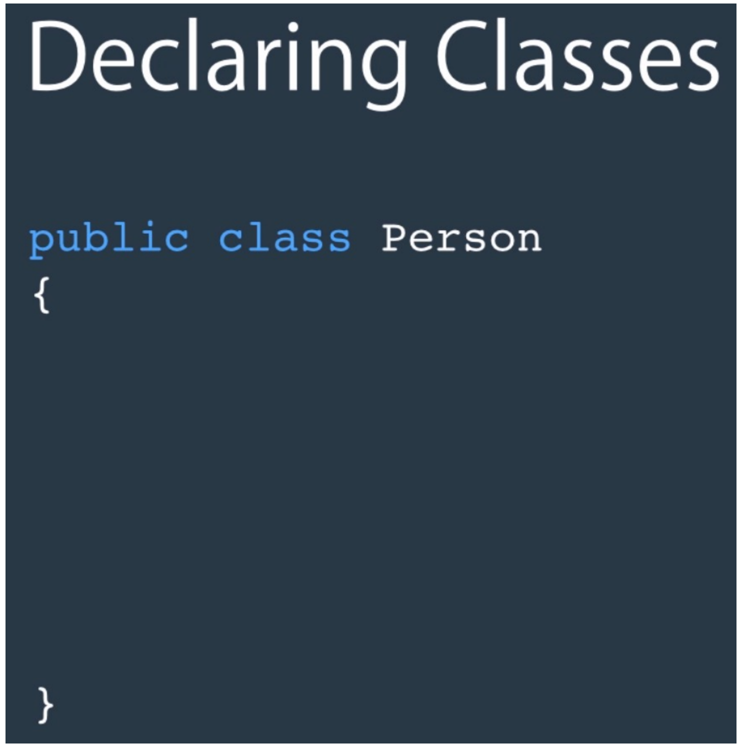
Class - is a type or a blueprint from which we create objects. Object is an instance of a class.



Classes -

More accurately when you run your application it's these objects that are talking to each other and collaborating to provide some functionality. But the world class and objects are often used interchangeably.

Creating a Class in C#


Declaring Classes

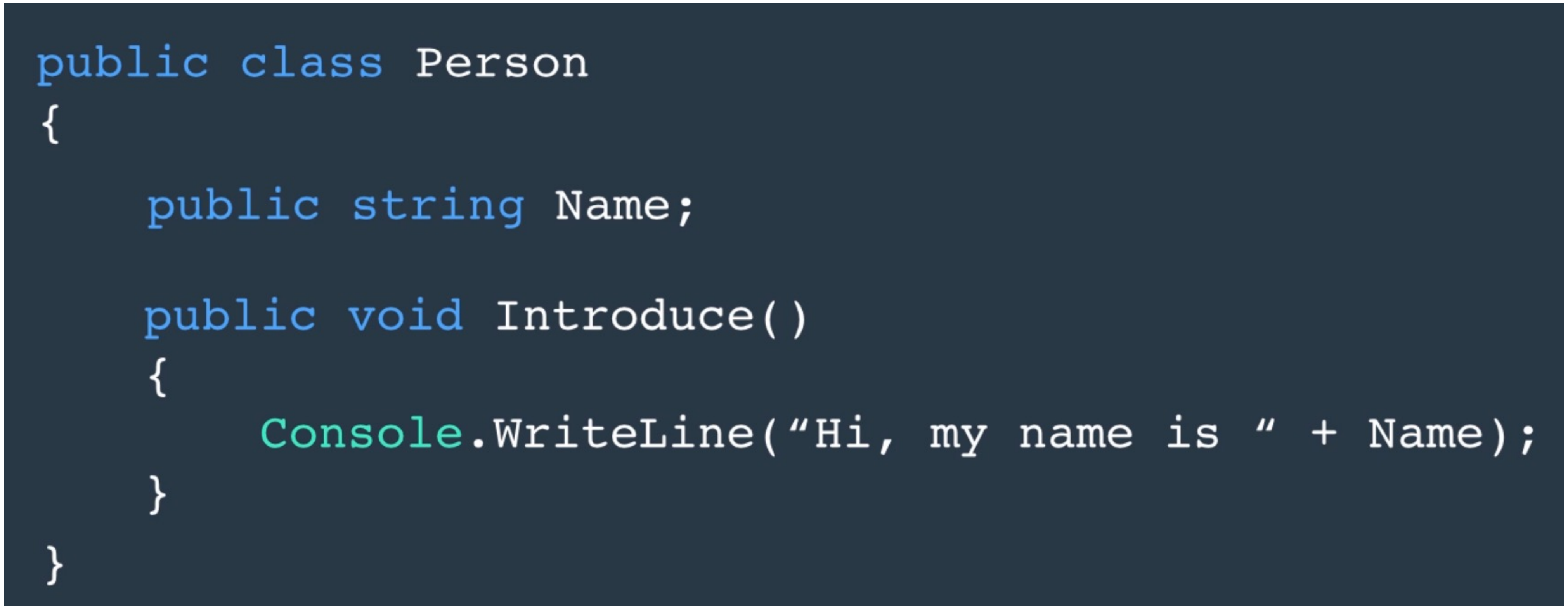
```
public class Person
{



}
```

public – Access Modifier
class – Keyword
Person - Identifier

Access modifier determines who can access this class.

Just remember whenever you want to create a class use the public keyword to make the class accessible anywhere in your application.

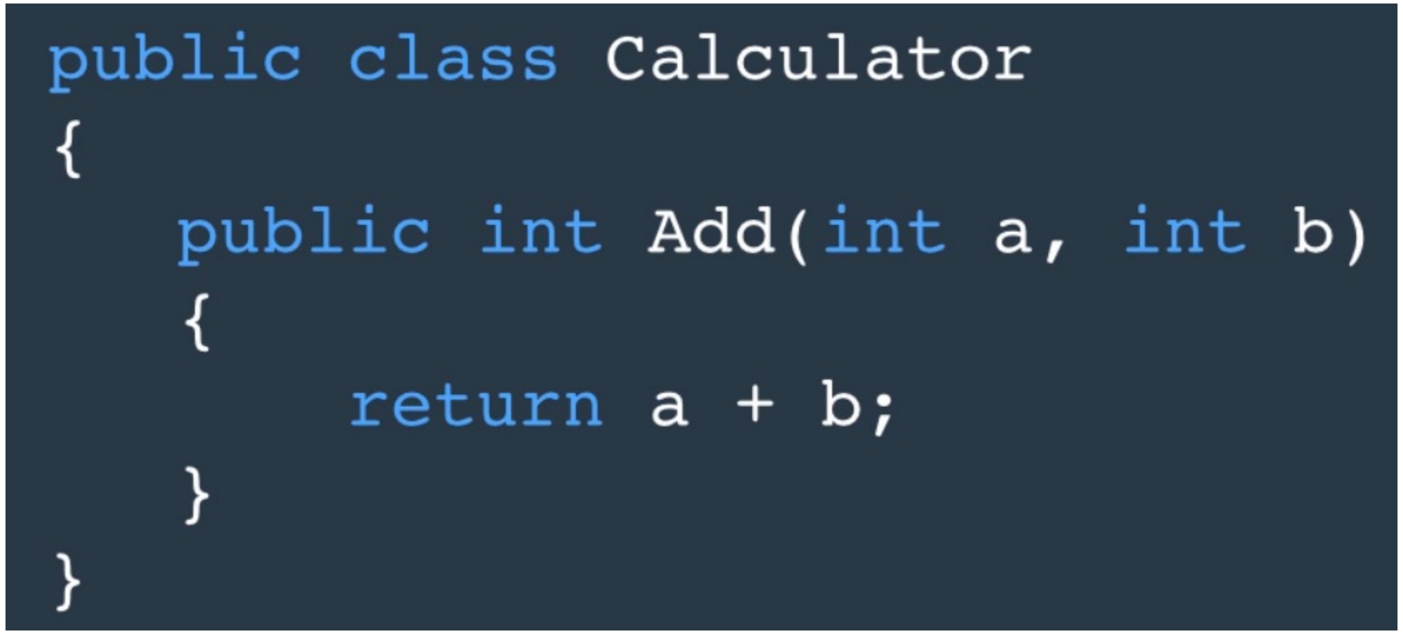
Creating a Class in C#



```
public class Person
{
    public string Name;

    public void Introduce()
    {
        Console.WriteLine(“Hi, my name is “ + Name);
    }
}
```

Method retun type - void
Method – Does not take any parameters

Creating a Class in C#



```
public class Calculator
{
    public int Add(int a, int b)
    {
        return a + b;
    }
}
```

Method retun type - int
Method – Take two any parameters

Creating Objects

```
int number;

Person person
```

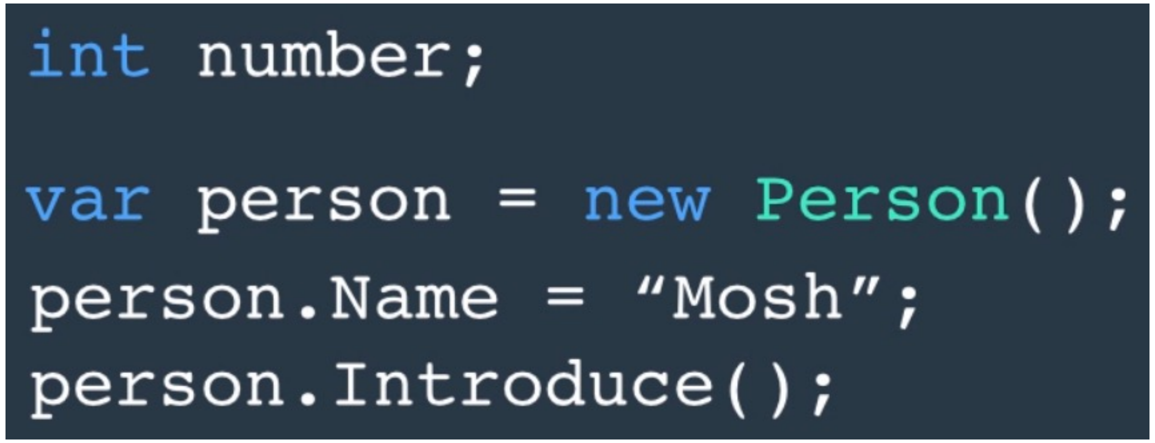
We need to allocate memory to the object by using “new” operator keyword.
C-sharp classes are treated differently than primitive types.

```
Person person = new Person();
```

We need to explicitly allocate memory for them. But the good thing is unlike languages like Objective C or C++ you do not have to worry about the allocating that memory. CLR our Common Language Runtime will take care of that for you. It has a process called garbage collection which automatically removes all objects that are not used.

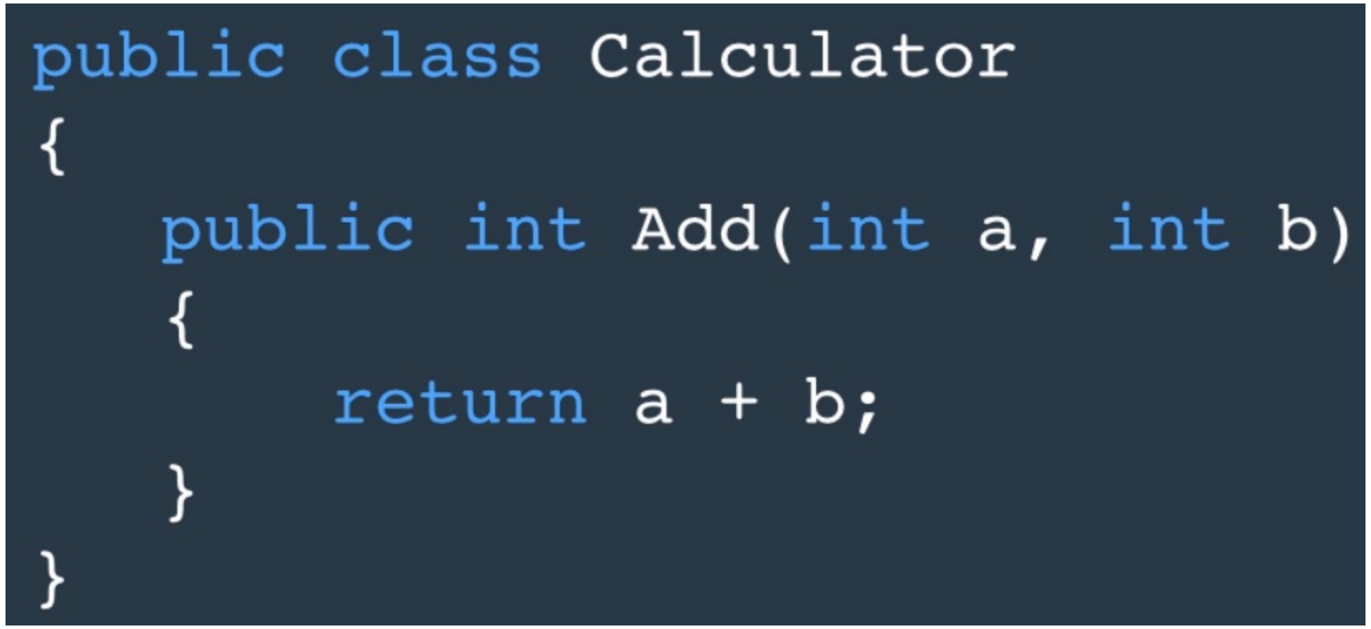
```
var person = new Person();
```

Classes – Dot Notation



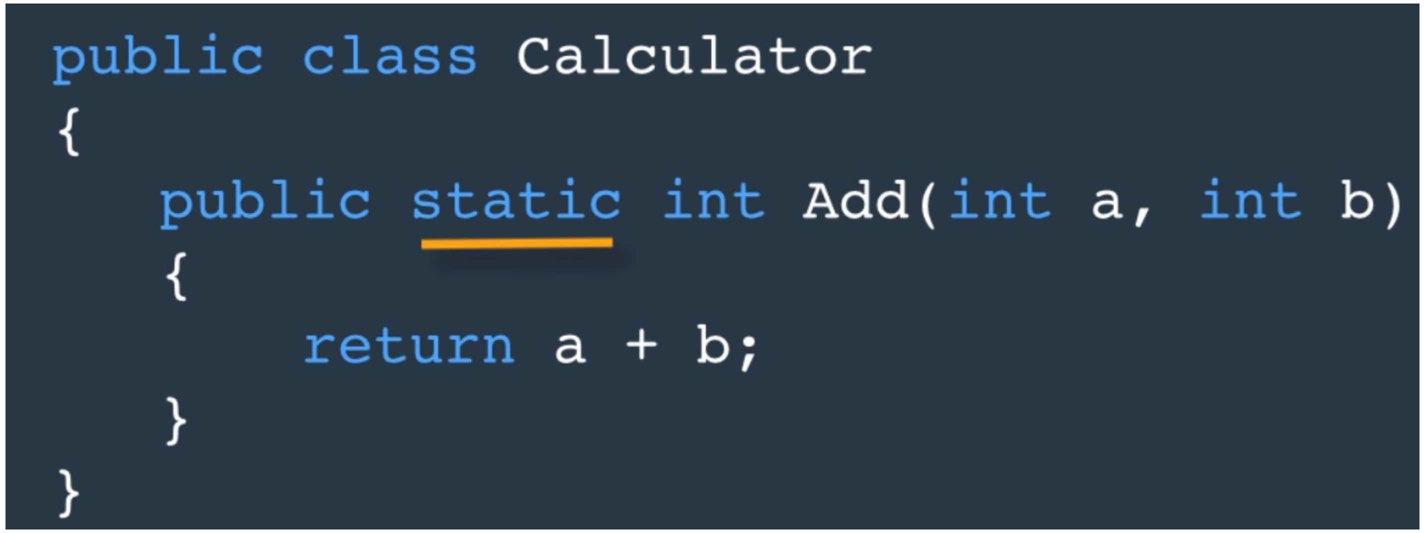
```
int number;

var person = new Person();
person.Name = “Mosh”;
person.Introduce();
```

Static Modifier

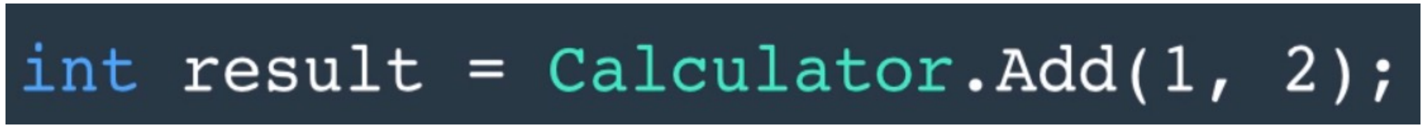


```
public class Calculator
{
    public int Add(int a, int b)
    {
        return a + b;
    }
}
```

Static Modifier



```
public class Calculator
{
    public static int Add(int a, int b)
    {
        return a + b;
    }
}
```

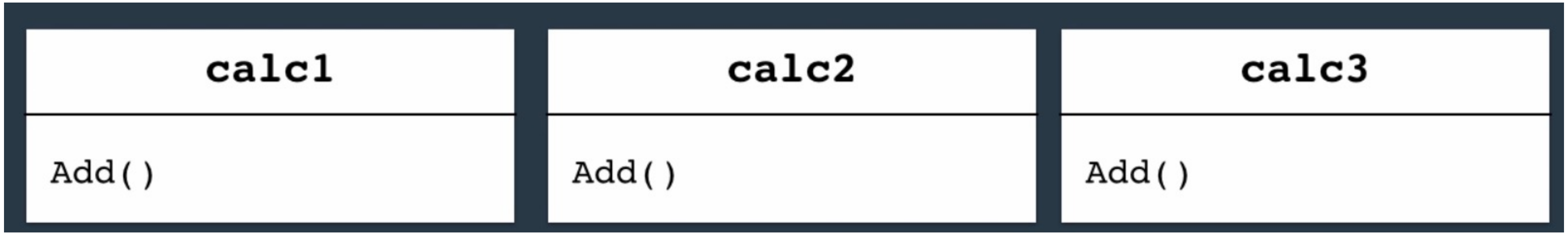


```
int result = Calculator.Add(1, 2);
```

A result we can access that method directly by the calculator class itself. We do not have to create an object to access a static member.

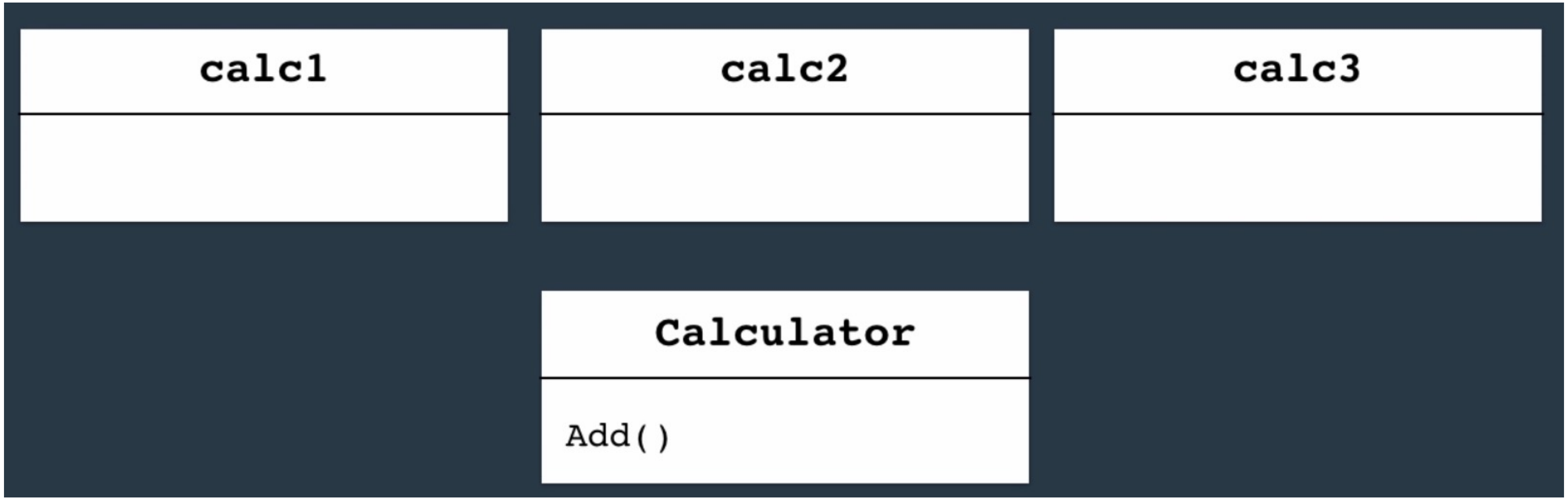
In fact we cannot access static members from objects.

Static Modifier



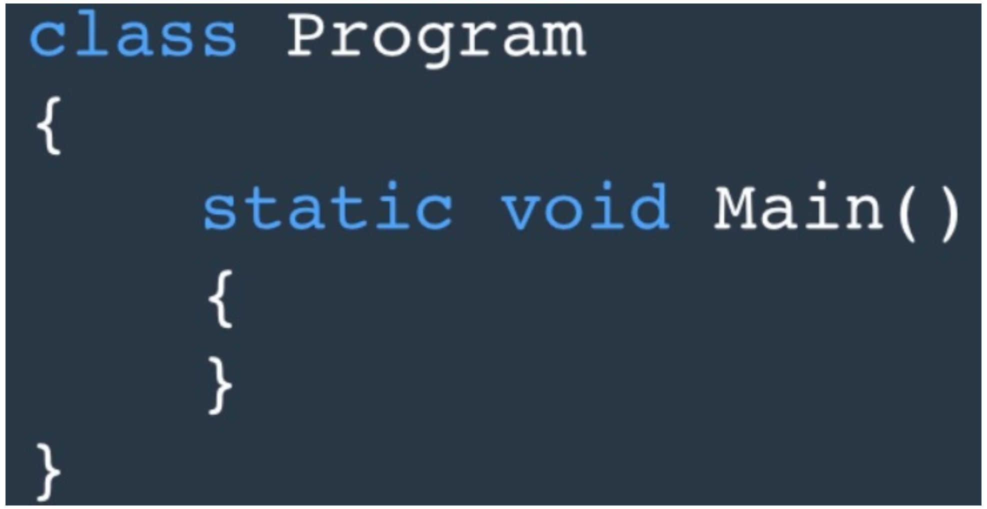
Without the static modifier when you create three objects of this calculator class each object in the memory will have the add method. But when you apply the static modifier the add method will be only in one place in memory and that is the calculator class itself. So it's not going to be repeated three times in memory.

Static Modifier



We use the static modifier when we want to present a concept that only a single instance of that should exist in memory.

Static Modifier – Examples



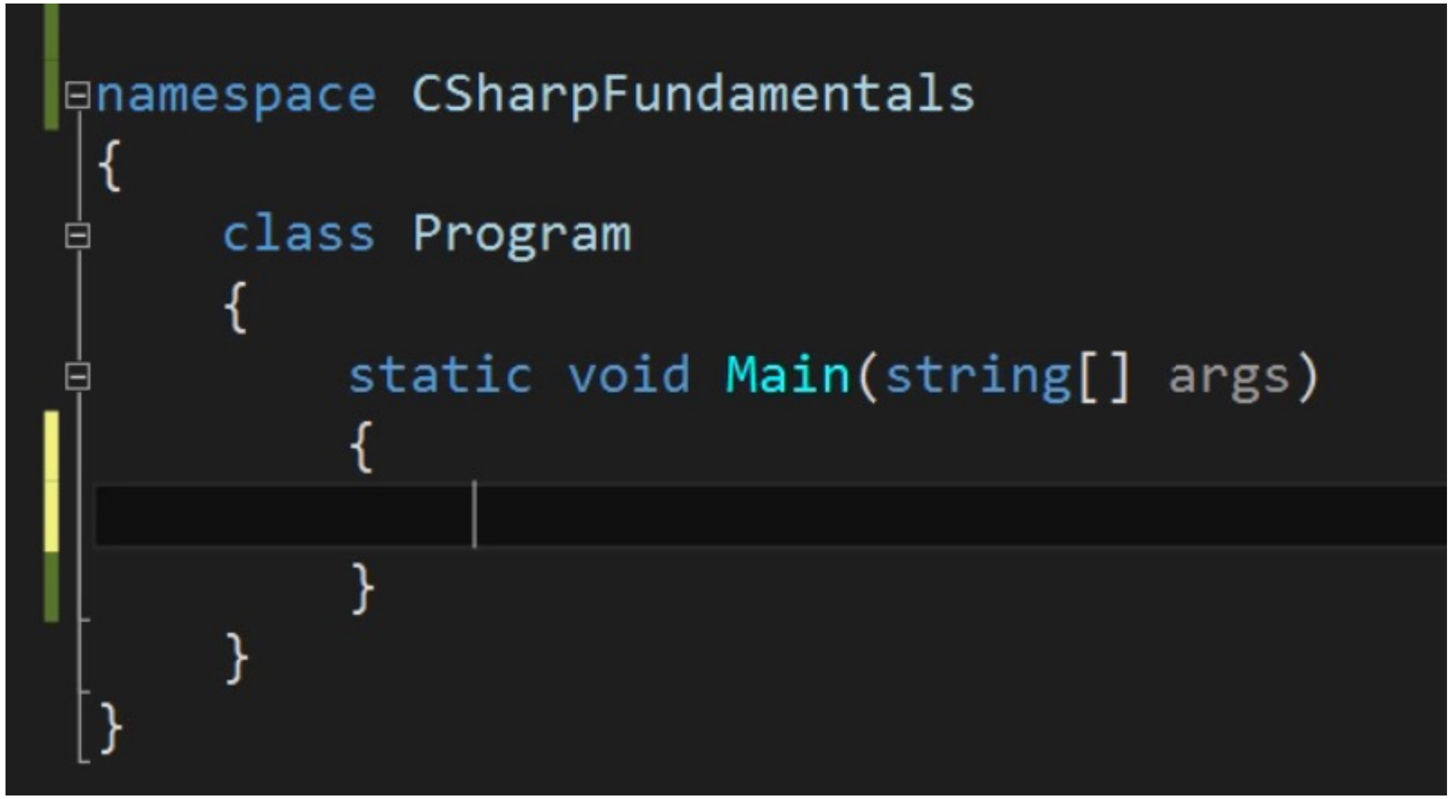
```
class Program
{
    static void Main()
    {
    }
}
```

Remember the Program class in our first program. We had a main method and Main was modified with the static keyword which means there is only one instance of the main method in memory. There is only one entry point in each C-Sharp application.

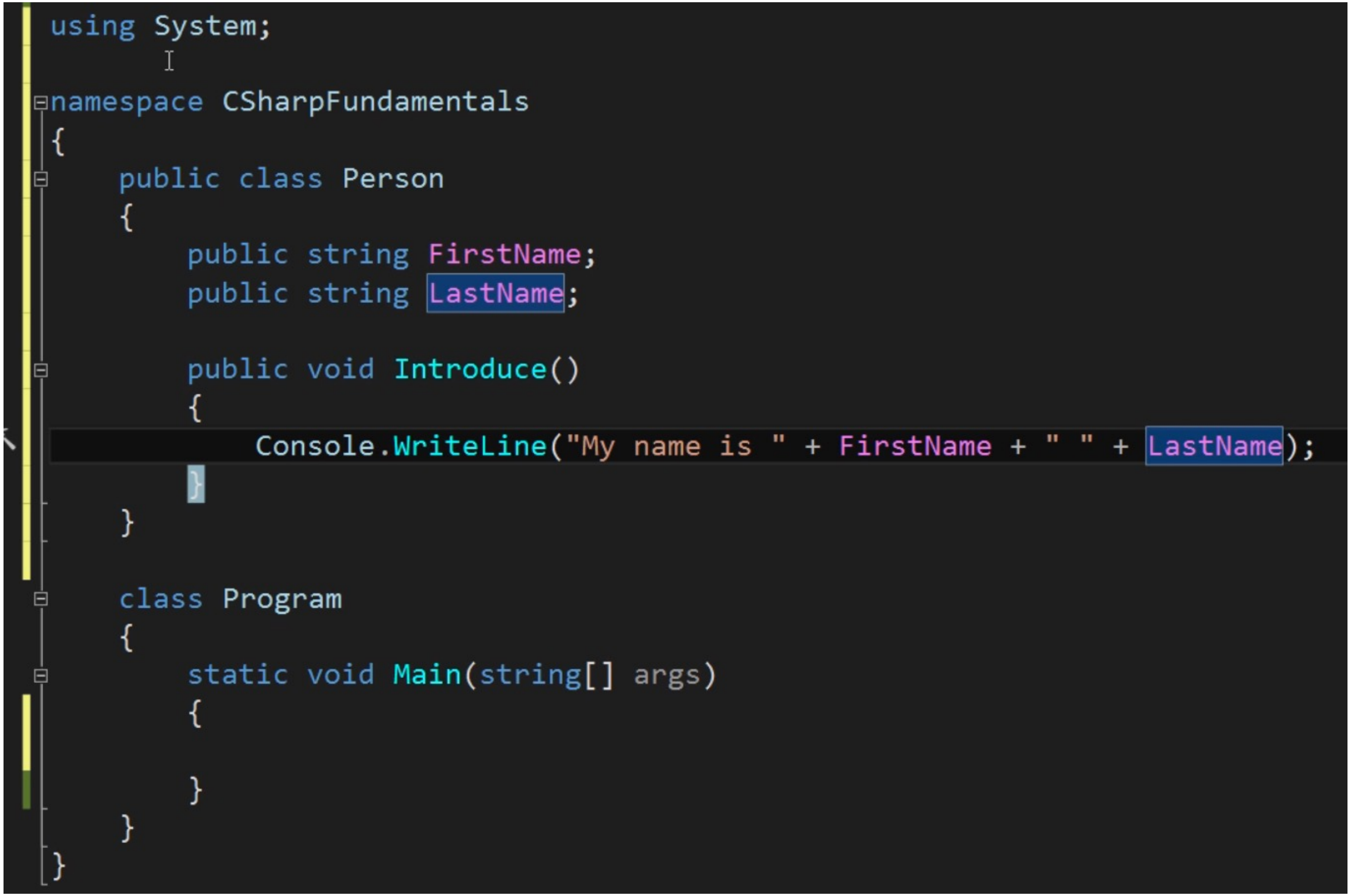
Current day time is another example. We don't want to have multiple date time objects in memory each representing a current date time. We want only one place in memory where you can look at the current data.

When we modify any members of a class whether it's a field or a method with a static modifier that member will be accessible from the class itself not an object

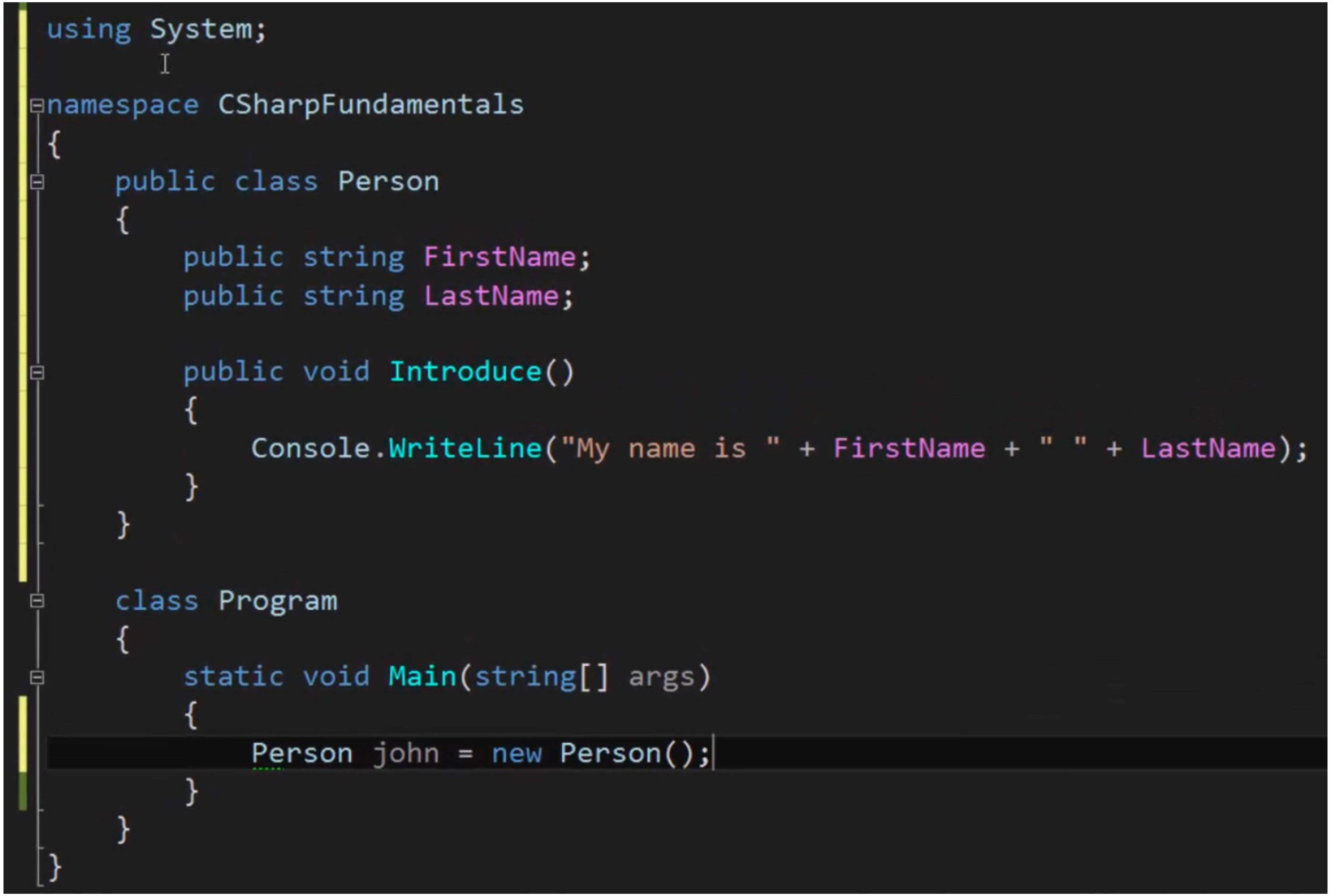
Visual Studio Demo- Classes



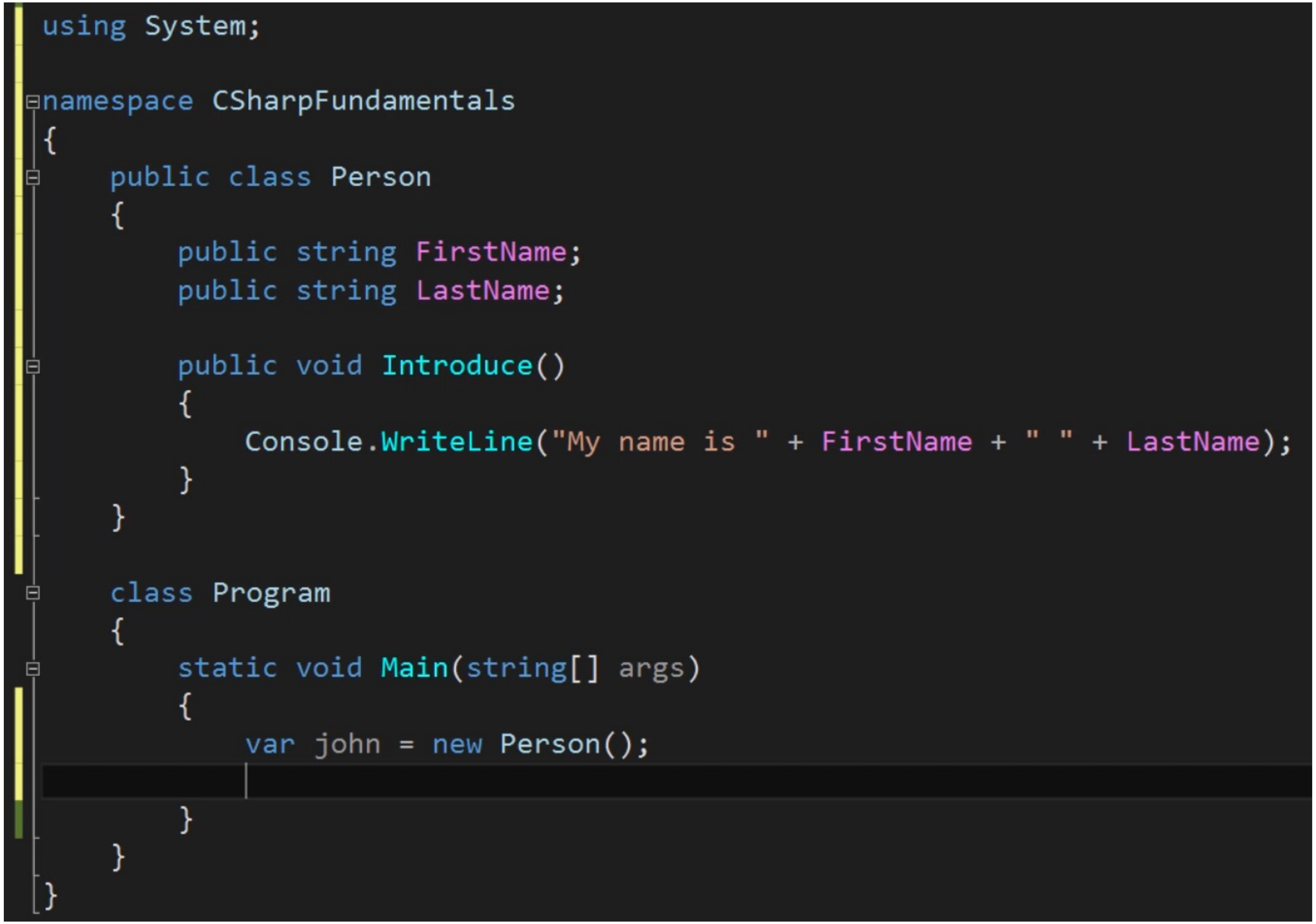
```
namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {

        }
    }
}
```



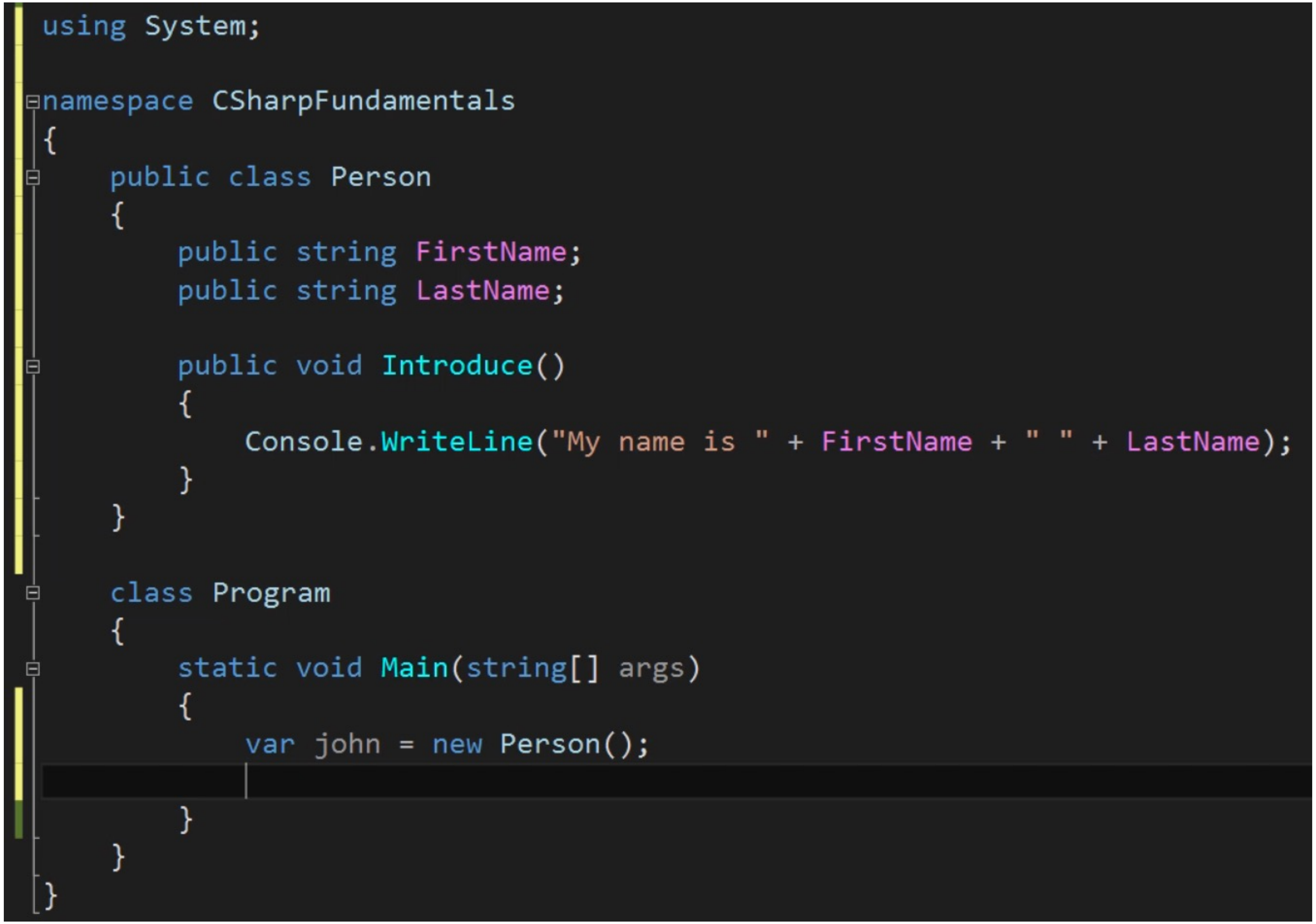
```
using System;

namespace CSharpFundamentals
{
    public class Person
    {
        public string FirstName;
        public string LastName;

        public void Introduce()
        {
            Console.WriteLine("My name is " + FirstName + " " + LastName);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {

        }
    }
}
```



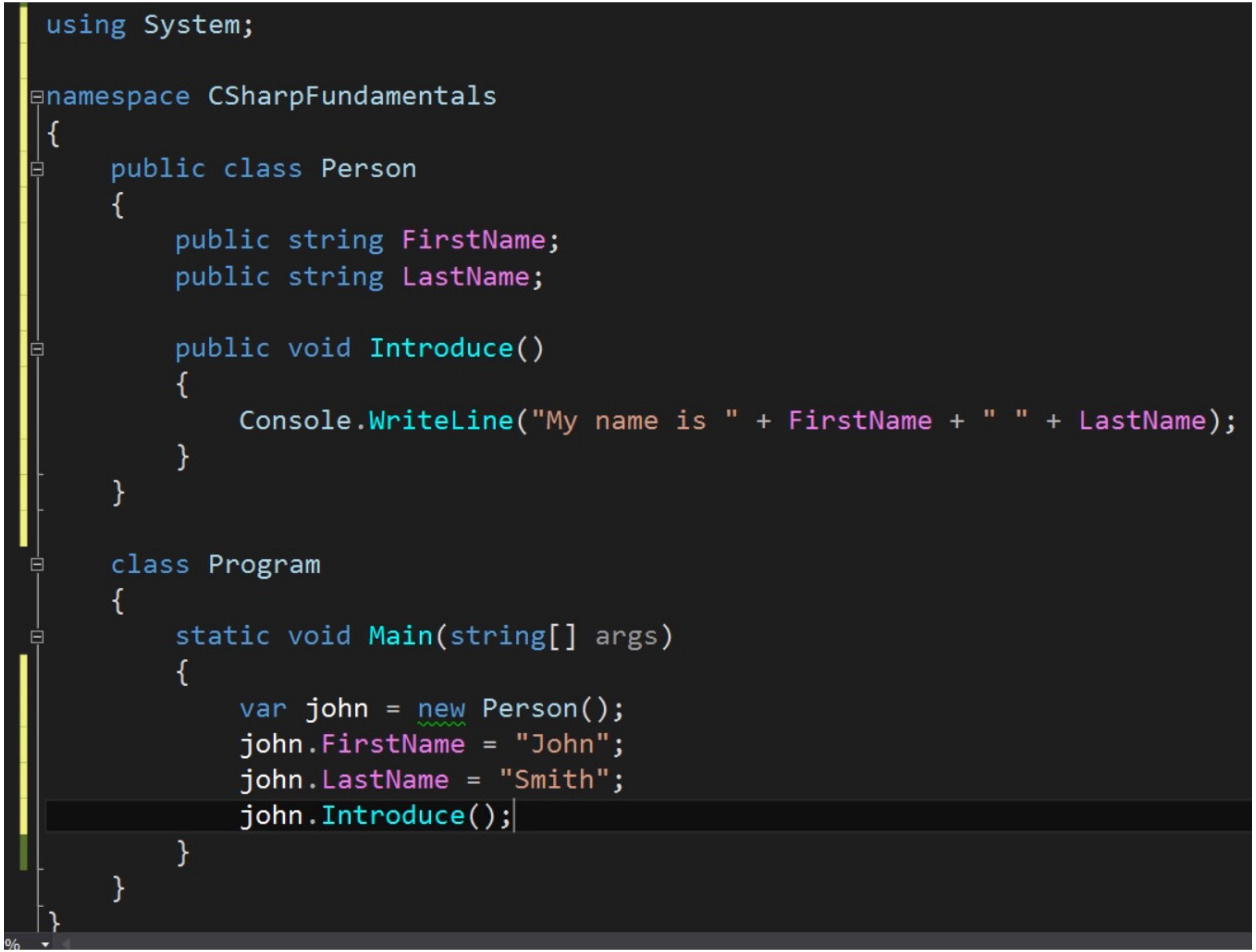
```
using System;

namespace CSharpFundamentals
{
    public class Person
    {
        public string FirstName;
        public string LastName;

        public void Introduce()
        {
            Console.WriteLine("My name is " + FirstName + " " + LastName);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Person john = new Person();
        }
    }
}
```



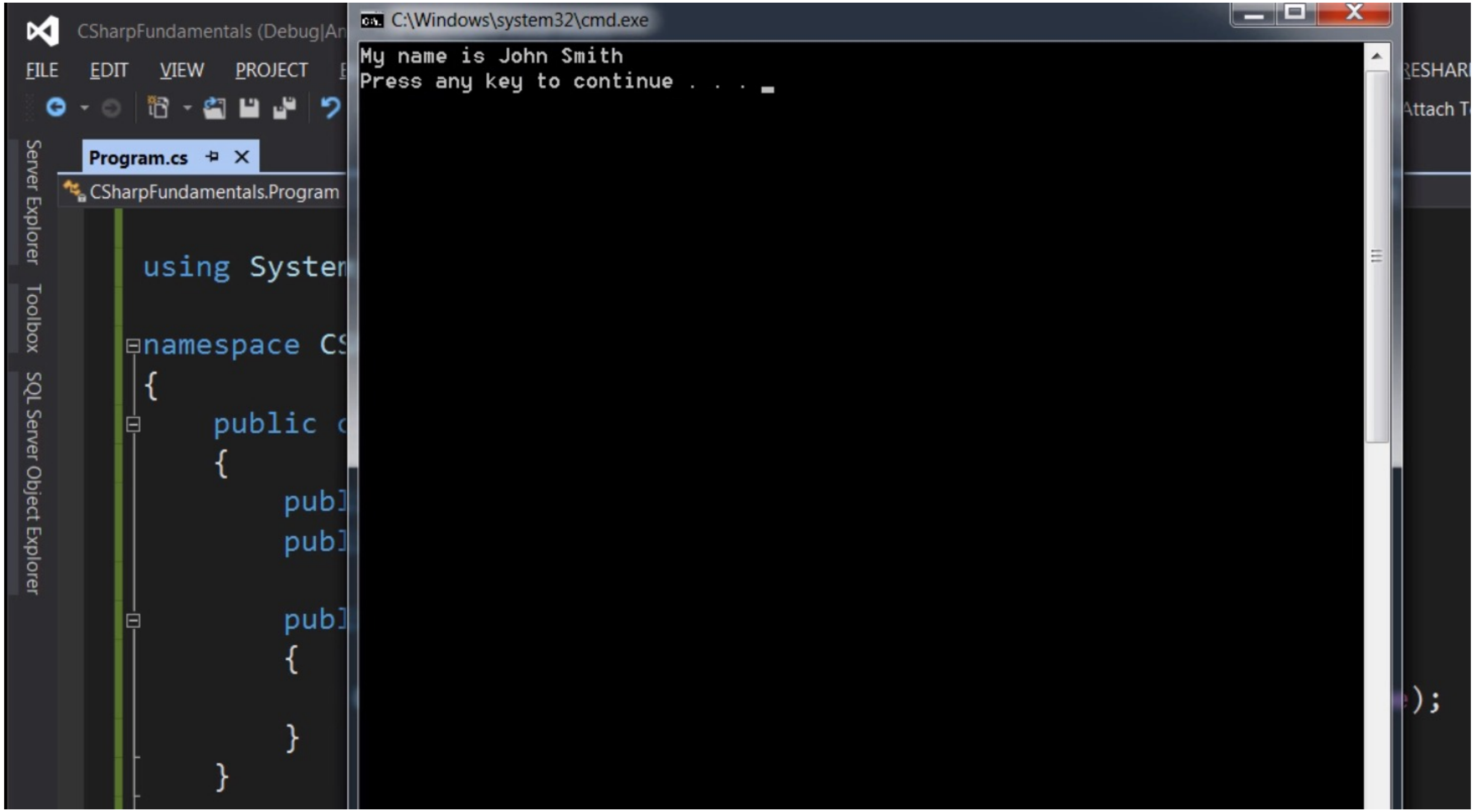
```
using System;

namespace CSharpFundamentals
{
    public class Person
    {
        public string FirstName;
        public string LastName;

        public void Introduce()
        {
            Console.WriteLine("My name is " + FirstName + " " + LastName);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            var john = new Person();
            
        }
    }
}
```



```csharp
using System;

namespace CSharpFundamentals
{
    public class Person
    {
        public string FirstName;
        public string LastName;

        public void Introduce()
        {
            Console.WriteLine("My name is " + FirstName + " " + LastName);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            var john = new Person();
            
        }
    }
}
```

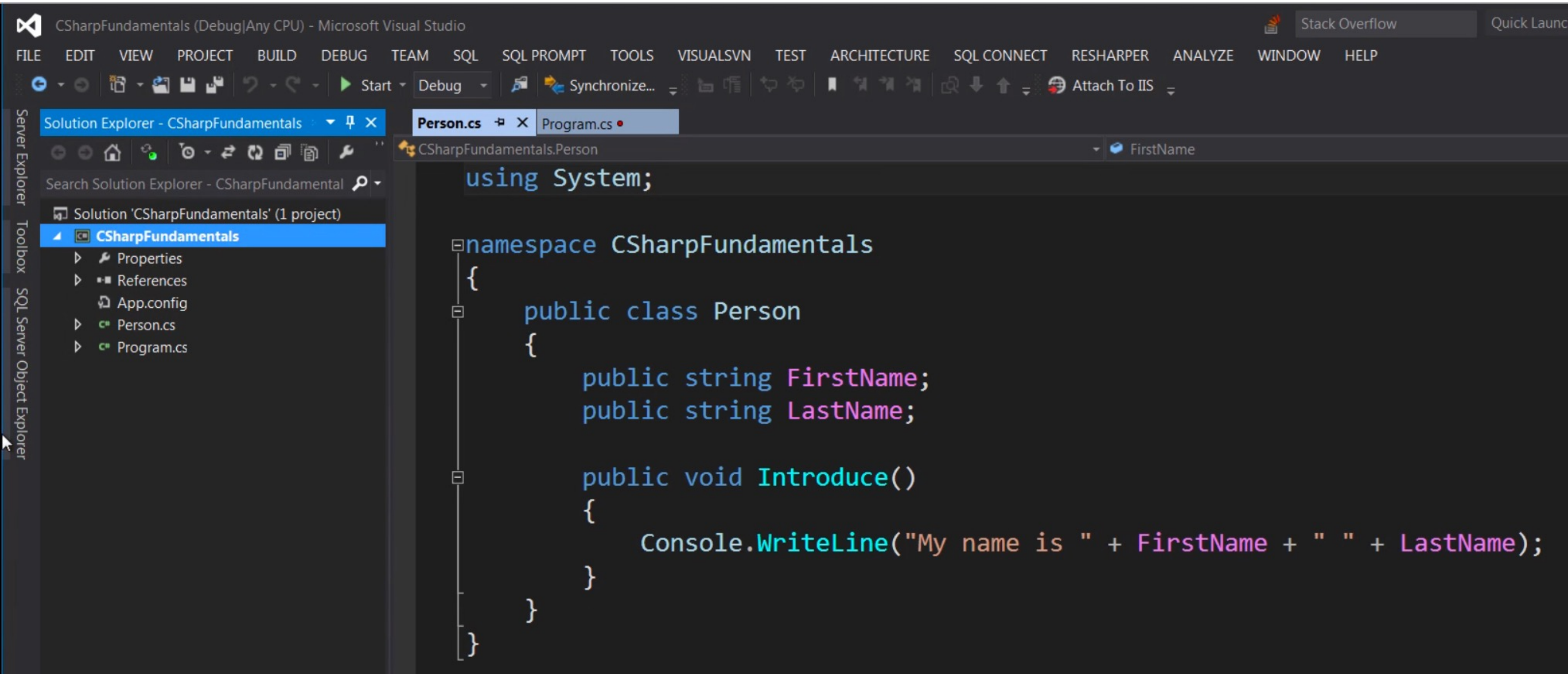


```
using System;

namespace CSharpFundamentals
{
    public class Person
    {
        public string FirstName;
        public string LastName;

        public void Introduce()
        {
            Console.WriteLine("My name is " + FirstName + " " + LastName);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            var john = new Person();
            john.FirstName = "John";
            john.LastName = "Smith";
            john.Introduce();
        }
    }
}
```

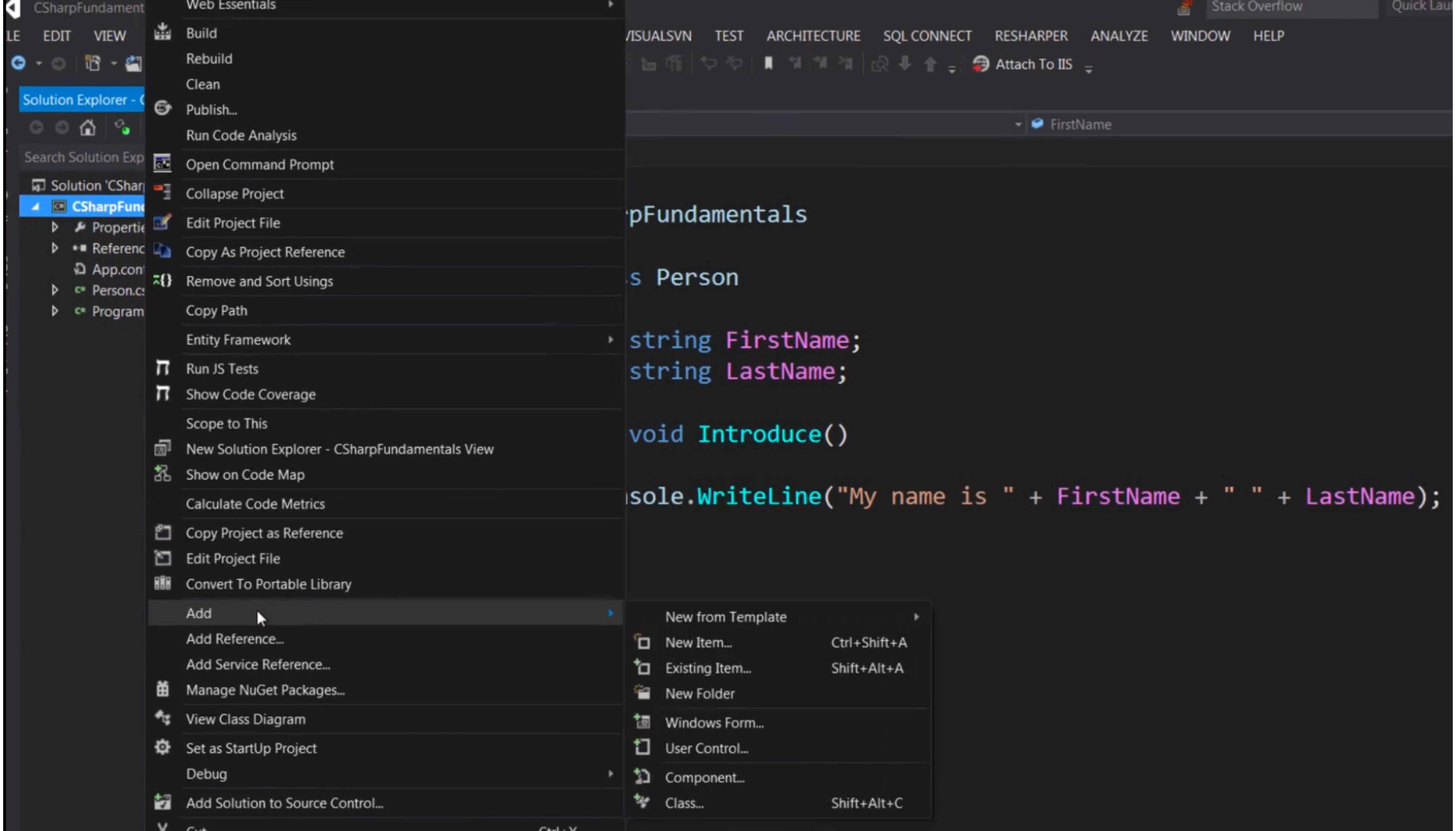

CSharpFundamentals (Debug|An
FILE EDIT VIEW PROJECT
Program.cs
CSharpFundamentals.Program
Server Explorer
Toolbox
SQL Server Object Explorer
C:\Windows\system32\cmd.exe
RESHAR
Attach T

```
My name is John Smith
Press any key to continue . . .
```

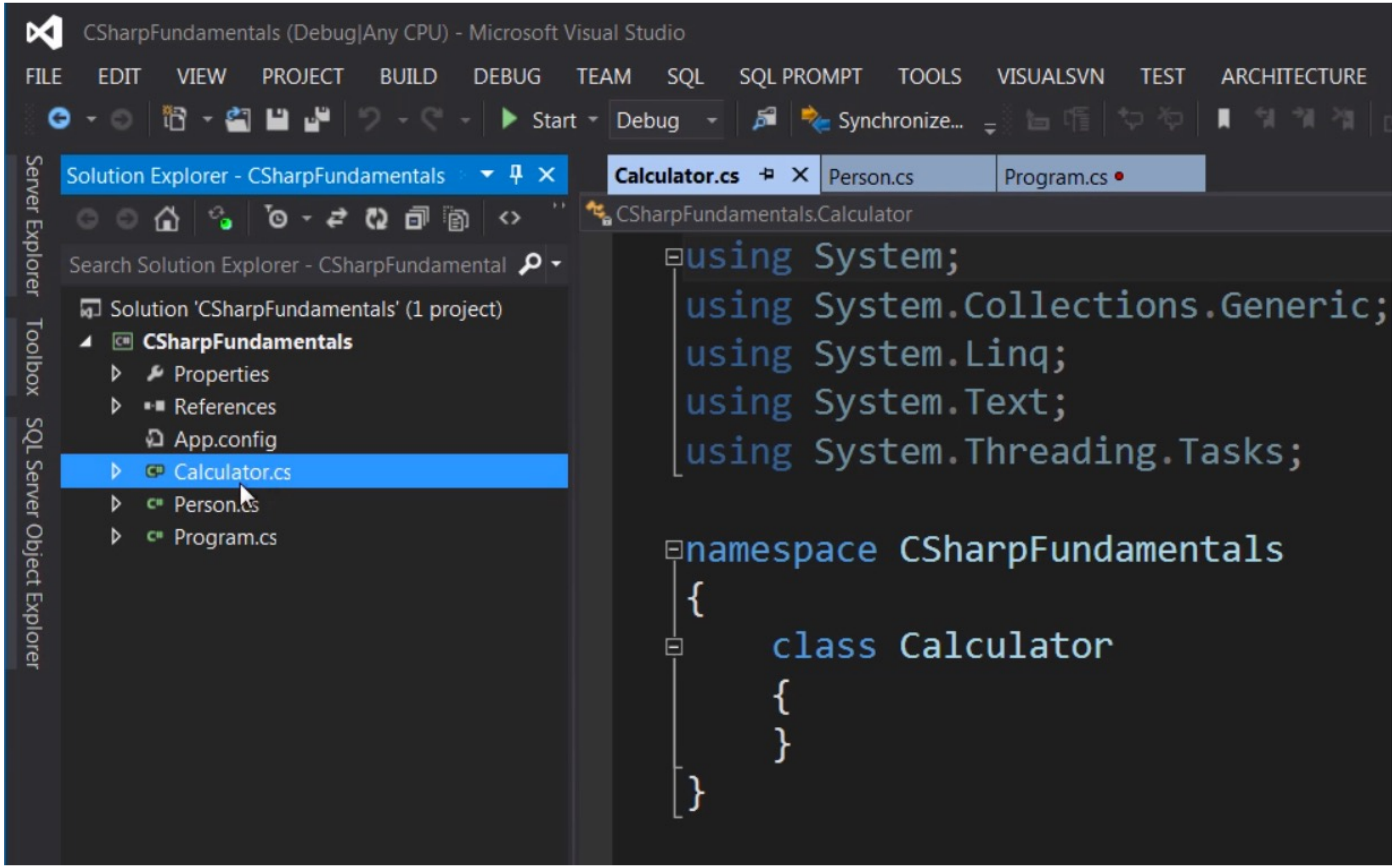
```
using Syster
namespace CS
{
    public c
    {
        publ
        publ

        publ
        {

        }
    }
```


CSharpFundamentals (Debug|Any CPU) - Microsoft Visual Studio
Stack Overflow
Quick Launc
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL SQL PROMPT TOOLS VISUALSVN TEST ARCHITECTURE SQL CONNECT RESHARPER ANALYZE WINDOW HELP
Start
Debug
Synchronize...
Attach To IIS
Server Explorer
Toolbox
SQL Server Object Explorer
Solution Explorer - CSharpFundamentals
Search Solution Explorer - CSharpFundamental
Solution 'CSharpFundamentals' (1 project)
CSharpFundamentals
Properties
References
App.config
Person.cs
Program.cs
Person.cs
Program.cs
CSharpFundamentals.Person
FirstName

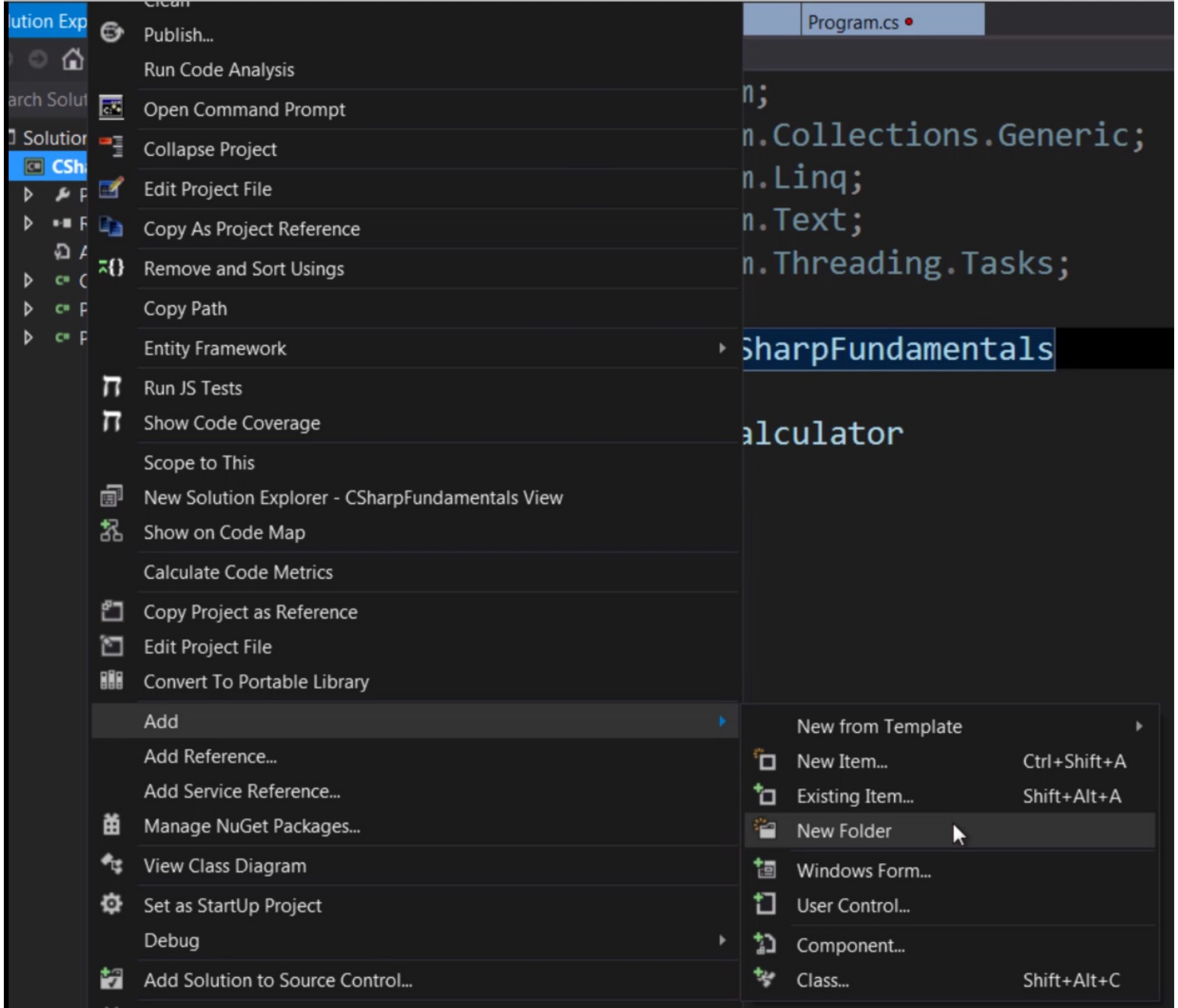
```csharp
using System;

namespace CSharpFundamentals
{
    public class Person
    {
        public string FirstName;
        public string LastName;

        public void Introduce()
        {
            Console.WriteLine("My name is " + FirstName + " " + LastName);
        }
    }
}
```



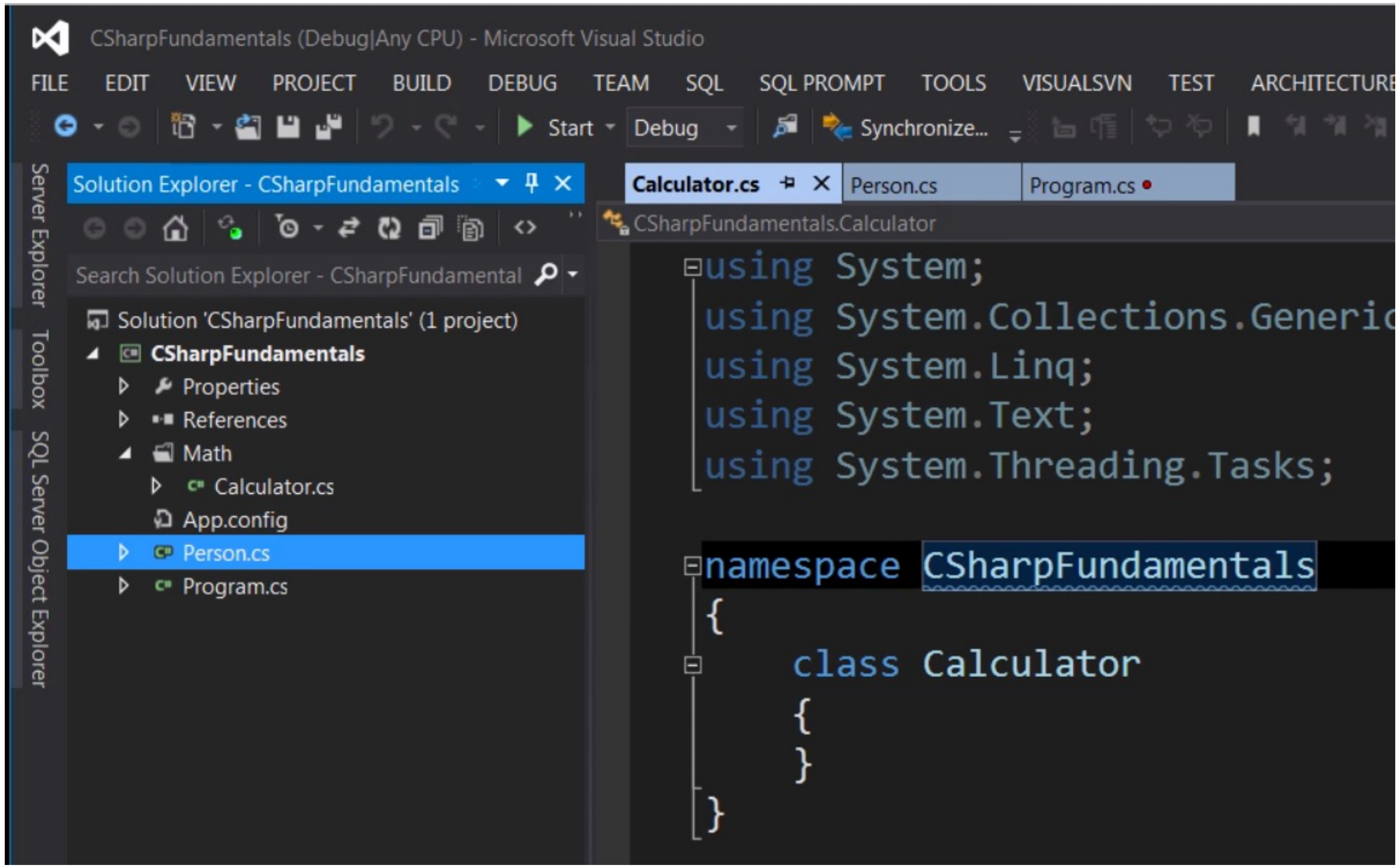
```
rpFundamentals

s Person

string FirstName;
string LastName;

void Introduce()

nsole.WriteLine("My name is " + FirstName + " " + LastName);
```


CSharpFundamentals (Debug|Any CPU) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL SQL PROMPT TOOLS VISUALSVN TEST ARCHITECTURE
Start
Debug
Synchronize...
Server Explorer
Toolbox
SQL Server Object Explorer
Solution Explorer - CSharpFundamentals
Search Solution Explorer - CSharpFundamental
Solution 'CSharpFundamentals' (1 project)
CSharpFundamentals
Properties
References
App.config
Calculator.cs
Person.cs
Program.cs
Calculator.cs
Person.cs
Program.cs
CSharpFundamentals.Calculator

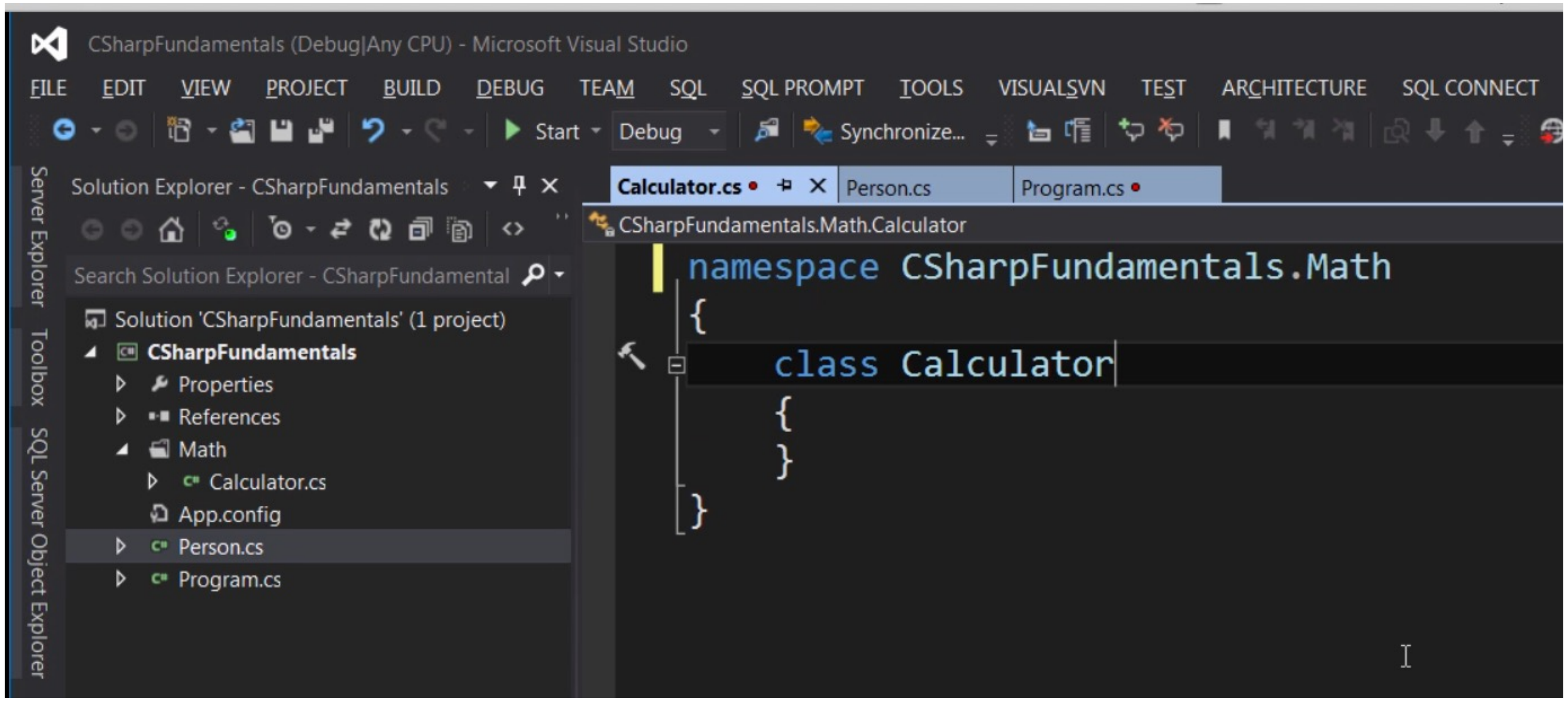
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CSharpFundamentals
{
    class Calculator
    {
    }
}
```



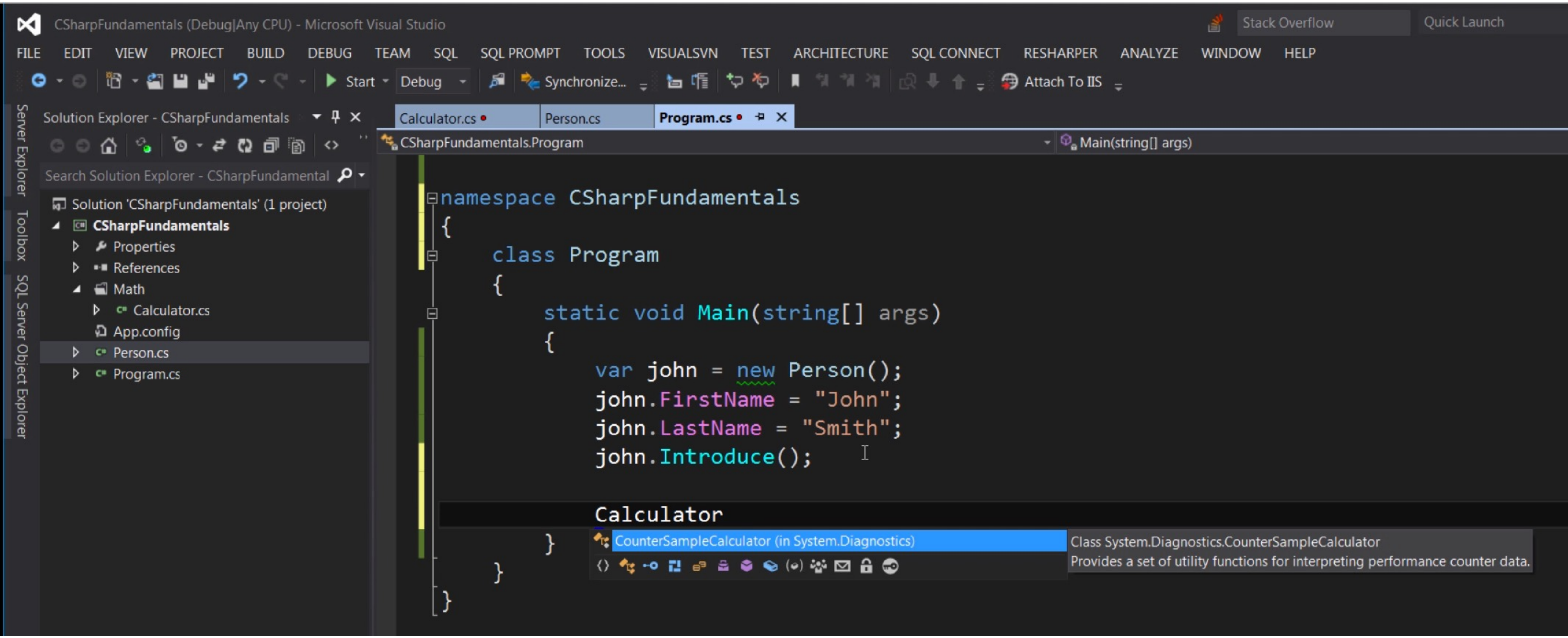
```
n;
n.Collections.Generic;
n.Linq;
n.Text;
n.Threading.Tasks;

SharpFundamentals

alculator
```


CSharpFundamentals (Debug|Any CPU) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL SQL PROMPT TOOLS VISUALSVN TEST ARCHITECTURE
Start Debug Synchronize...
Solution Explorer - CSharpFundamentals
Search Solution Explorer - CSharpFundamental
Solution 'CSharpFundamentals' (1 project)
CSharpFundamentals
Properties
References
Math
Calculator.cs
App.config
Person.cs
Program.cs
Server Explorer
Toolbox
SQL Server Object Explorer
Calculator.cs
Person.cs
Program.cs
CSharpFundamentals.Calculator

```
using System;
using System.Collections.Generic
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CSharpFundamentals
{
    class Calculator
    {
    }
}
```



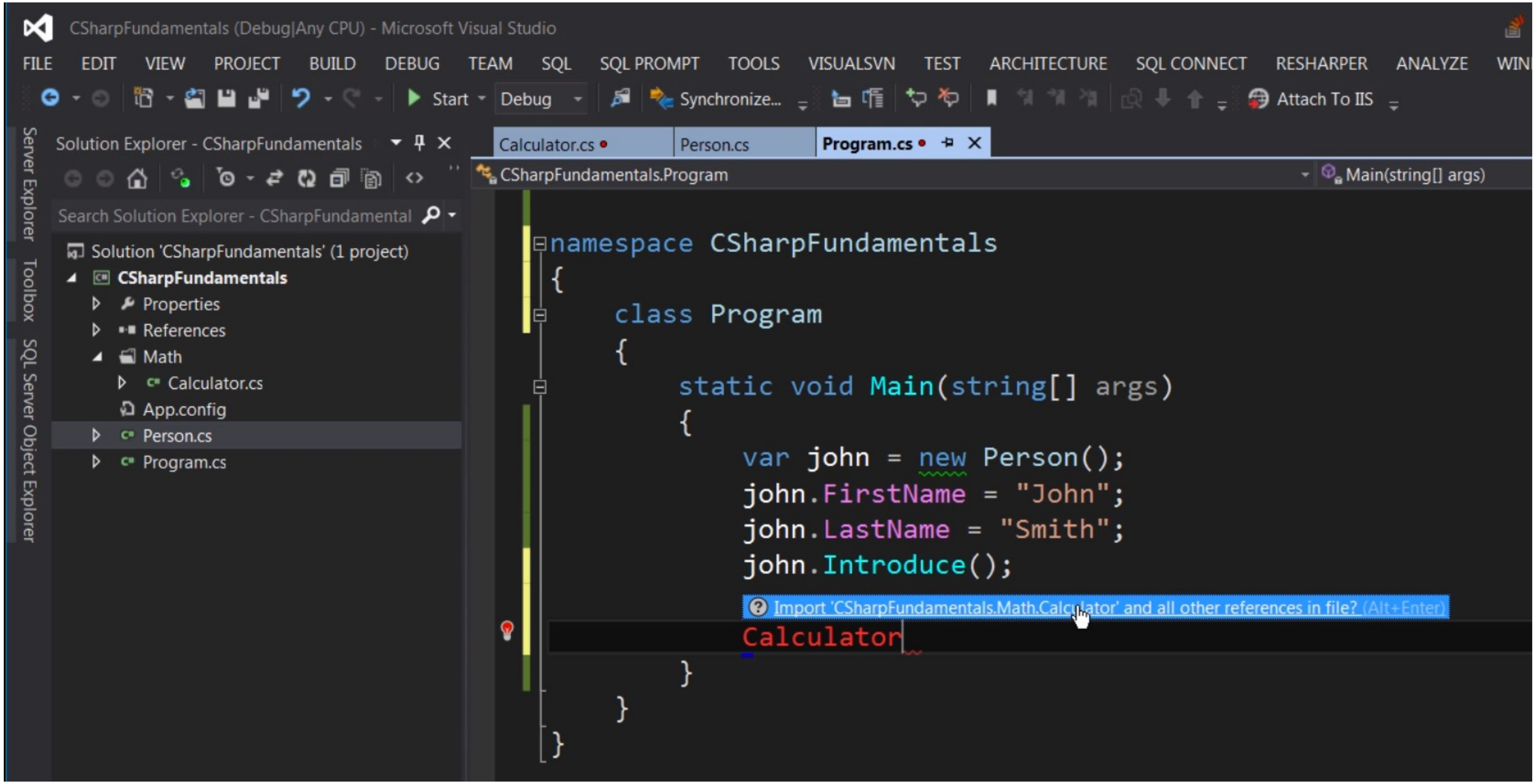
```
namespace CSharpFundamentals.Math
{
    class Calculator
    {
    }
}
```


CSharpFundamentals (Debug|Any CPU) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL SQL PROMPT TOOLS VISUALSVN TEST ARCHITECTURE SQL CONNECT RESHARPER ANALYZE WINDOW HELP
Start Debug Synchronize... Attach To IIS
Solution Explorer - CSharpFundamentals
Solution 'CSharpFundamentals' (1 project)
CSharpFundamentals
Properties
References
Math
Calculator.cs
App.config
Person.cs
Program.cs
Calculator.cs
Person.cs
Program.cs
CSharpFundamentals.Program
Main(string[] args)

```
namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var john = new Person();
            john.FirstName = "John";
            john.LastName = "Smith";
            john.Introduce();

            Calculator
        }
    }
}
```

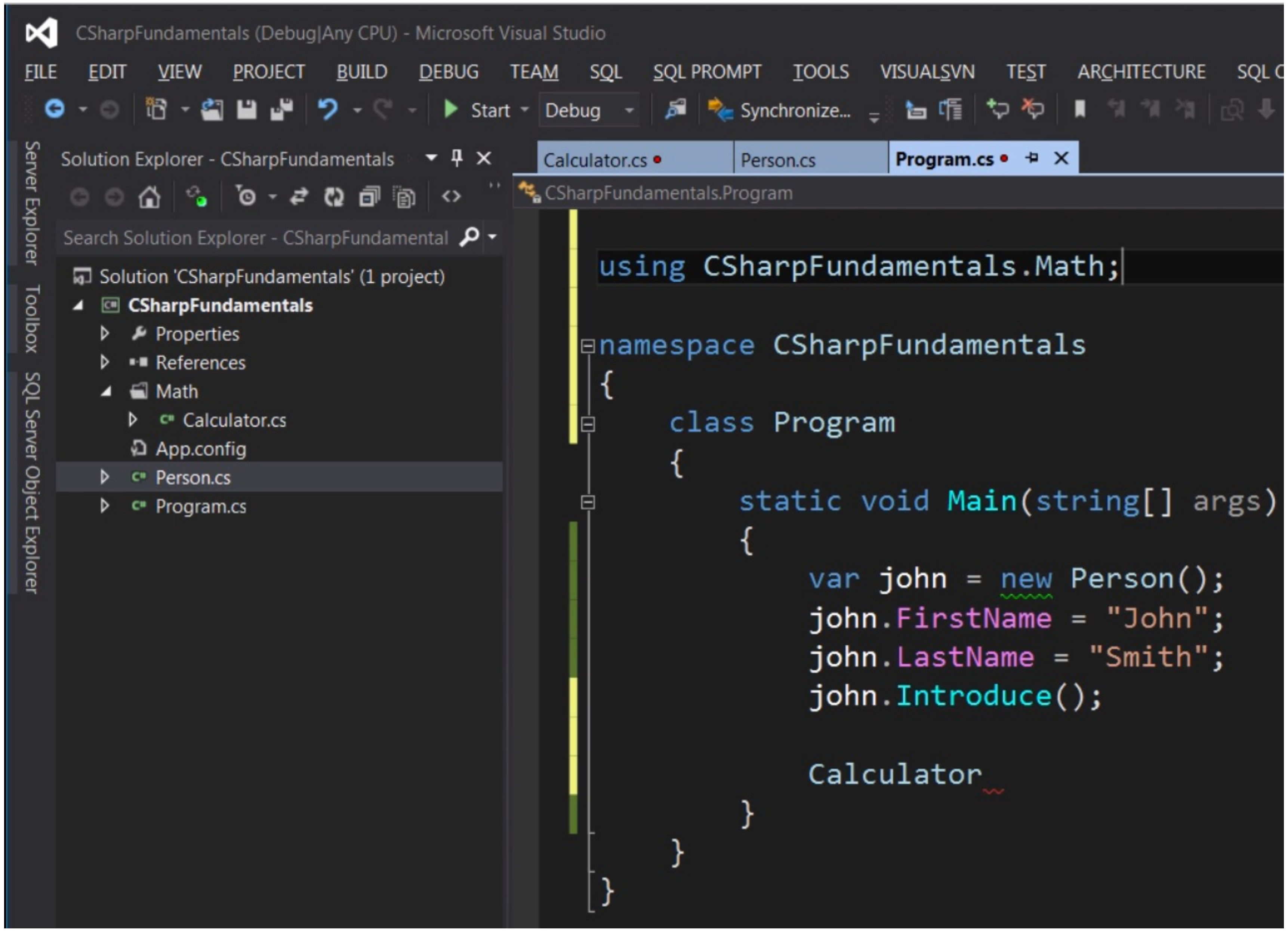
CounterSampleCalculator (in System.Diagnostics)

Class System.Diagnostics.CounterSampleCalculator
Provides a set of utility functions for interpreting performance counter data.

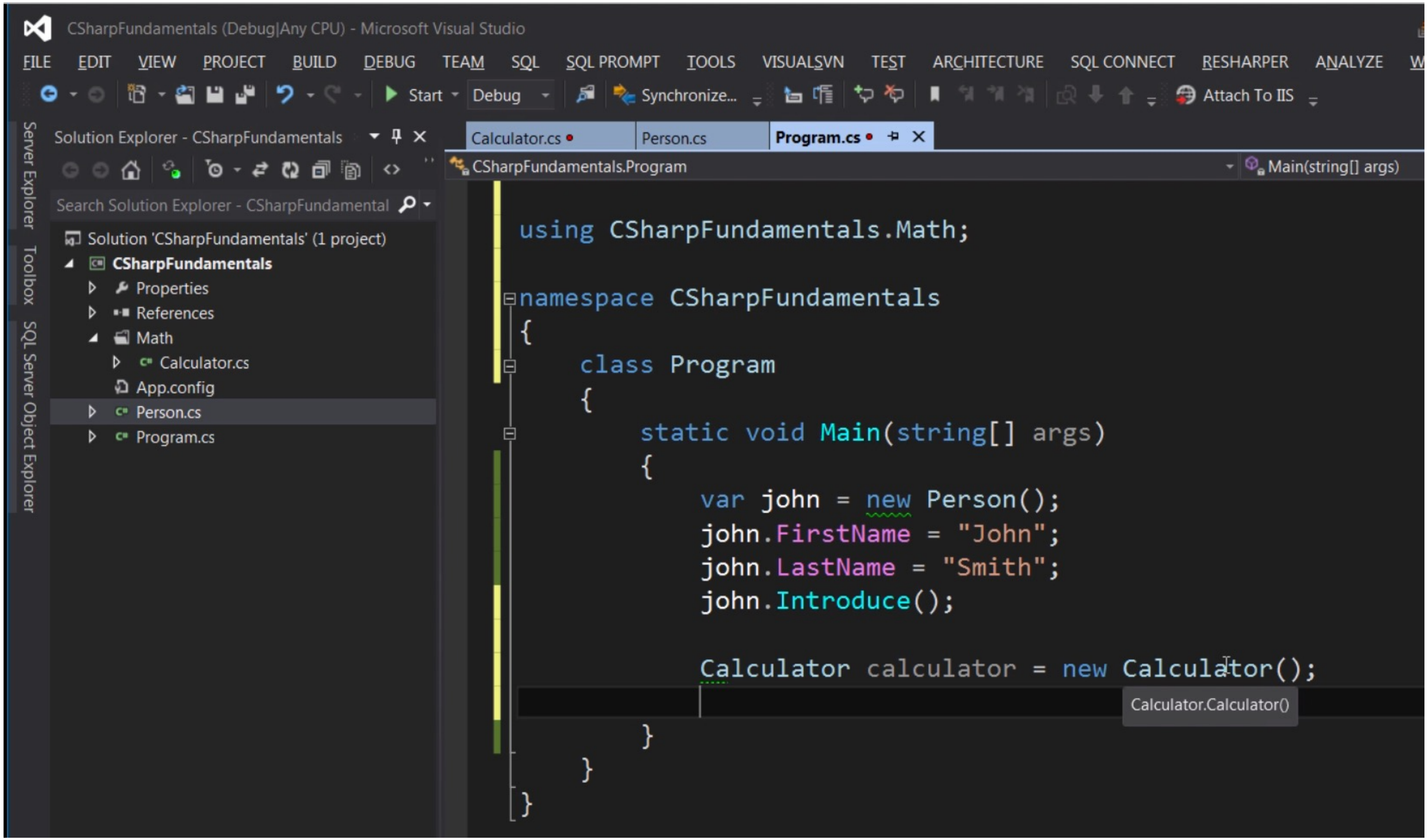

CSharpFundamentals (Debug|Any CPU) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL SQL PROMPT TOOLS VISUALSVN TEST ARCHITECTURE SQL CONNECT RESHARPER ANALYZE WIN
Start Debug Synchronize... Attach To IIS
Solution Explorer - CSharpFundamentals
Search Solution Explorer - CSharpFundamental
Solution 'CSharpFundamentals' (1 project)
CSharpFundamentals
Properties
References
Math
Calculator.cs
App.config
Person.cs
Program.cs
Calculator.cs
Person.cs
Program.cs
CSharpFundamentals.Program
Main(string[] args)
Server Explorer
Toolbox
SQL Server Object Explorer

```
namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var john = new Person();
            john.FirstName = "John";
            john.LastName = "Smith";
            john.Introduce();

            Calculator
        }
    }
}
```

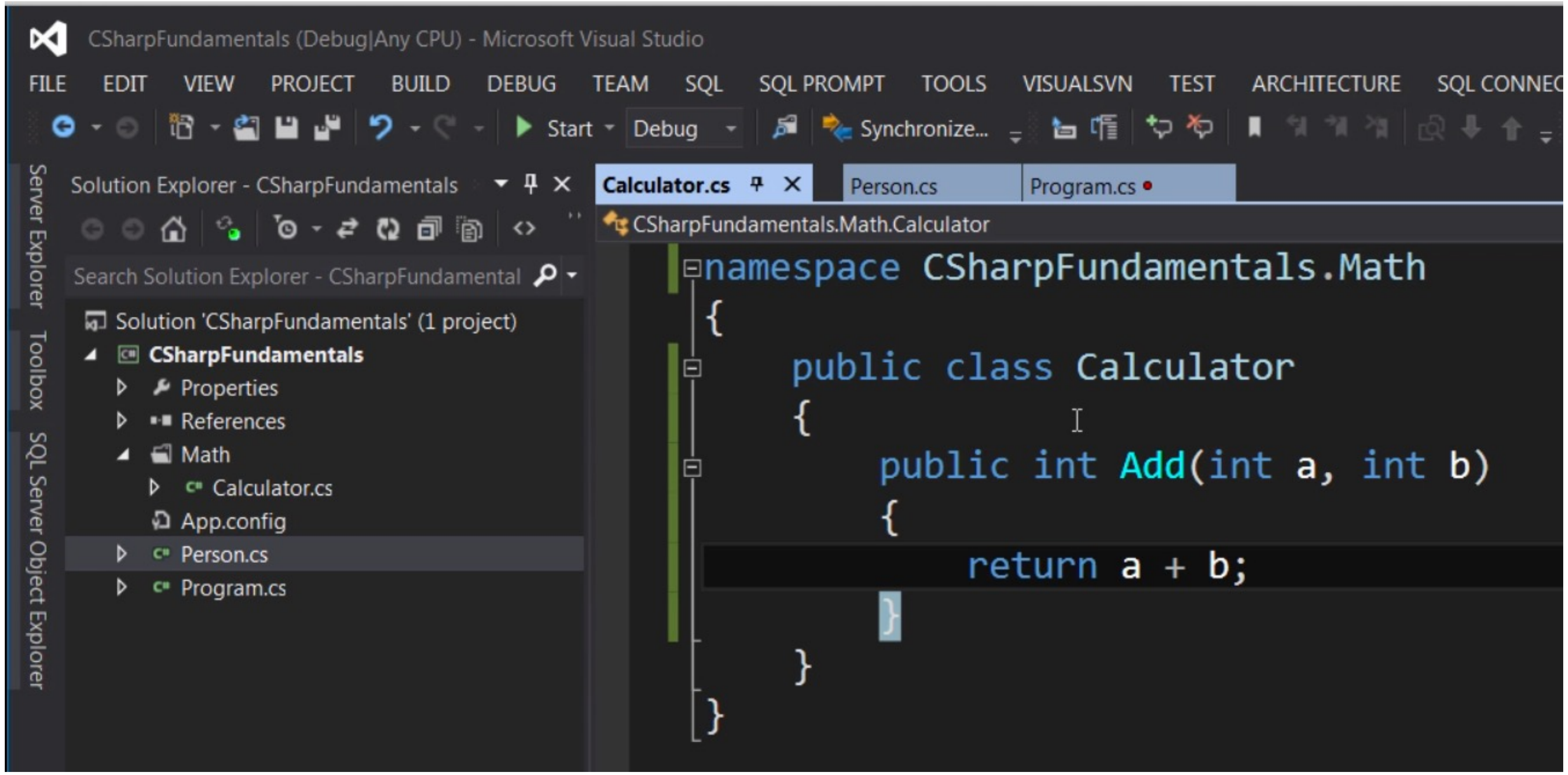
Import 'CSharpFundamentals.Math.Calculator' and all other references in file? (Alt+Enter)


CSharpFundamentals (Debug|Any CPU) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL SQL PROMPT TOOLS VISUALSVN TEST ARCHITECTURE
Start
Debug
Synchronize...
Server Explorer
Toolbox
SQL Server Object Explorer
Solution Explorer - CSharpFundamentals
Search Solution Explorer - CSharpFundamental
Solution 'CSharpFundamentals' (1 project)
CSharpFundamentals
Properties
References
Math
Calculator.cs
App.config
Person.cs
Program.cs
Calculator.cs
Person.cs
Program.cs
CSharpFundamentals.Program

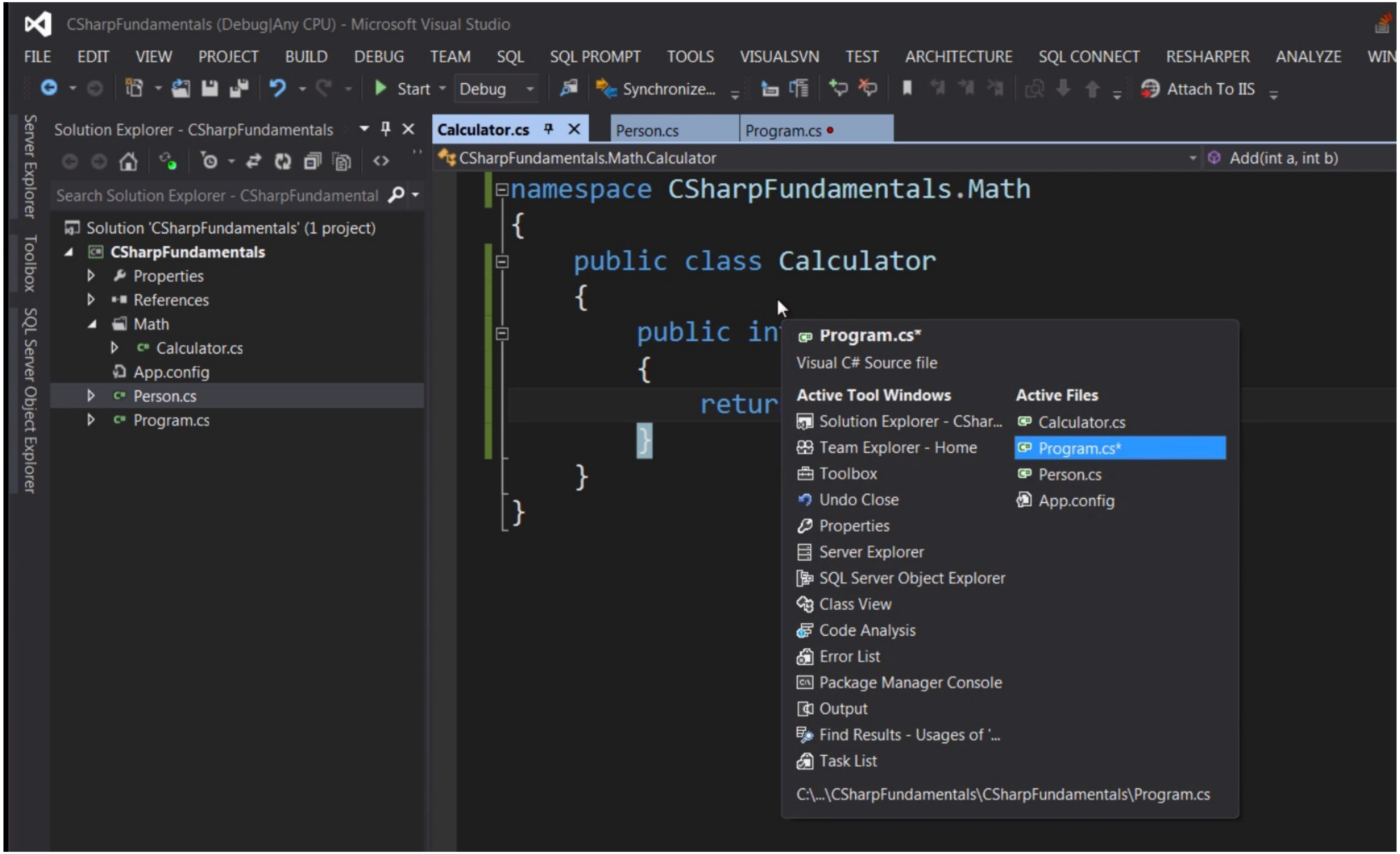
```
using CSharpFundamentals.Math;

namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var john = new Person();
            john.FirstName = "John";
            john.LastName = "Smith";
            john.Introduce();

            Calculator
        }
    }
}
```


CSharpFundamentals (Debug|Any CPU) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL SQL PROMPT TOOLS VISUALSVN TEST ARCHITECTURE SQL CONNECT RESHARPER ANALYZE
Start
Debug
Synchronize...
Attach To IIS
Server Explorer
Toolbox
SQL Server Object Explorer
Solution Explorer - CSharpFundamentals
Search Solution Explorer - CSharpFundamental
Solution 'CSharpFundamentals' (1 project)
CSharpFundamentals
Properties
References
Math
Calculator.cs
App.config
Person.cs
Program.cs
Calculator.cs
Person.cs
Program.cs
CSharpFundamentals.Program
Main(string[] args)
Calculator.Calculator()

```
using CSharpFundamentals.Math;

namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var john = new Person();
            john.FirstName = "John";
            john.LastName = "Smith";
            john.Introduce();

            Calculator calculator = new Calculator();

        }
    }
}
```


CSharpFundamentals (Debug|Any CPU) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL SQL PROMPT TOOLS VISUALSVN TEST ARCHITECTURE SQL CONNE
Start
Debug
Synchronize...
Solution Explorer - CSharpFundamentals
Search Solution Explorer - CSharpFundamental
Solution 'CSharpFundamentals' (1 project)
CSharpFundamentals
Properties
References
Math
Calculator.cs
App.config
Person.cs
Program.cs
Calculator.cs
Person.cs
Program.cs
CSharpFundamentals.Math.Calculator
Server Explorer
Toolbox
SQL Server Object Explorer

```
namespace CSharpFundamentals.Math
{
    public class Calculator
    {
        public int Add(int a, int b)
        {
            return a + b;
        }
    }
}
```



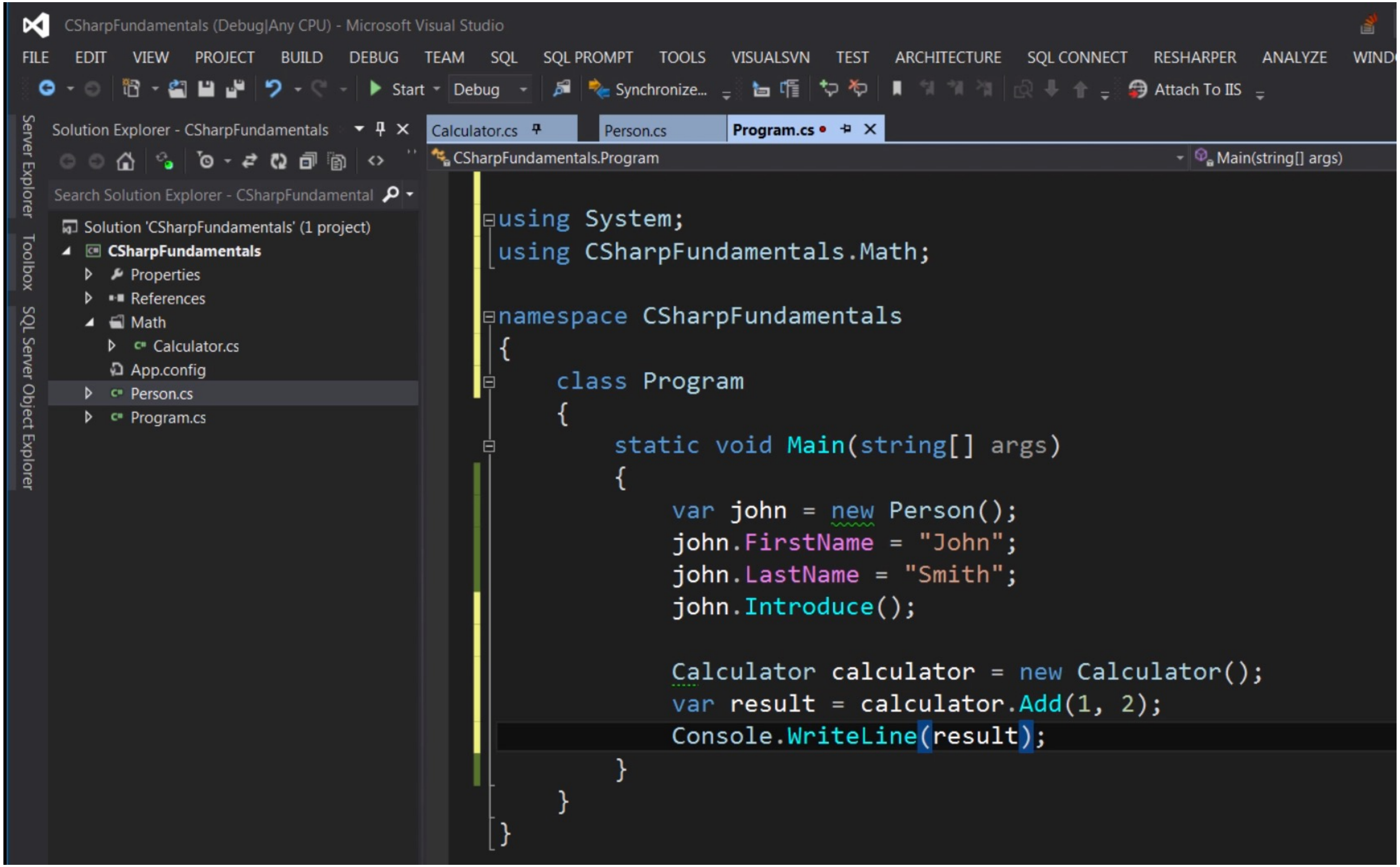
CSharpFundamentals (Debug|Any CPU) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL SQL PROMPT TOOLS VISUALSVN TEST ARCHITECTURE SQL CONNECT RESHARPER ANALYZE WIN
Start
Debug
Synchronize...
Attach To IIS
Solution Explorer - CSharpFundamentals
Search Solution Explorer - CSharpFundamental
Solution 'CSharpFundamentals' (1 project)
CSharpFundamentals
Properties
References
Math
Calculator.cs
App.config
Person.cs
Program.cs
Calculator.cs
Person.cs
Program.cs
CSharpFundamentals.Math.Calculator
Add(int a, int b)
Server Explorer
Toolbox
SQL Server Object Explorer

```
namespace CSharpFundamentals.Math
{
    public class Calculator
    {
        public in
        {
            retur
        }
    }
}
```

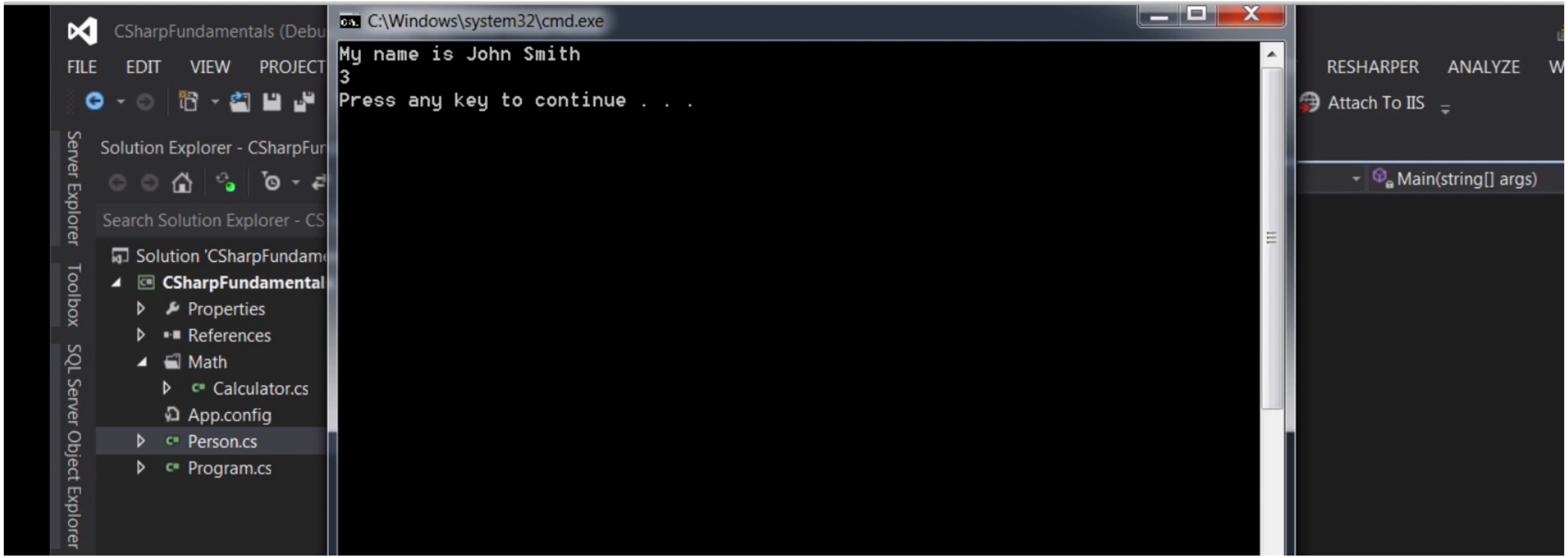
Program.cs*
Visual C# Source file

Active Tool Windows	Active Files
Solution Explorer - CShar...	Calculator.cs
Team Explorer - Home	Program.cs*
Toolbox	Person.cs
Undo Close	App.config
Properties	
Server Explorer	
SQL Server Object Explorer	
Class View	
Code Analysis	
Error List	
Package Manager Console	
Output	
Find Results - Usages of '...	
Task List	

C:\...\CSharpFundamentals\CSharpFundamentals\Program.cs


CSharpFundamentals (Debug|Any CPU) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL SQL PROMPT TOOLS VISUALSVN TEST ARCHITECTURE SQL CONNECT RESHARPER ANALYZE WIND
Start
Debug
Synchronize...
Attach To IIS
Server Explorer
Toolbox
SQL Server Object Explorer
Solution Explorer - CSharpFundamentals
Search Solution Explorer - CSharpFundamental
Solution 'CSharpFundamentals' (1 project)
CSharpFundamentals
Properties
References
Math
Calculator.cs
App.config
Person.cs
Program.cs
Calculator.cs
Person.cs
Program.cs
CSharpFundamentals.Program
Main(string[] args)

```
using System;
using CSharpFundamentals.Math;

namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var john = new Person();
            john.FirstName = "John";
            john.LastName = "Smith";
            john.Introduce();

            Calculator calculator = new Calculator();
            var result = calculator.Add(1, 2);
            Console.WriteLine(result);
        }
    }
}
```



```
My name is John Smith
3
Press any key to continue . . .
```