

Inheritance - Second Pillar of OOP



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Access Modifiers



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Agenda

- Why access modifiers are important
- Black box metaphor
- C# access modifiers



Black Box



Black Box



Access Modifiers

- public
- private
- protected
- internal
- protected internal



Public

Accessible from everywhere.

```
public class Customer
{
    public void Promote()
    {
    }
}
```

...

```
var customer = new Customer();
customer.Promote();
```



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Private

Accessible only from the class.

```
public class Customer
{
    private int CalculateRating(
    {
    }
}
```

...

```
var customer = new Customer();
customer.calculateRating();
```



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Protected

Accessible only from the class and its derived classes.

```
public class Customer
{
    protected int CalculateRating()
    {
    }
}
```

...

```
var customer = new Customer();
customer.calculateRating();
```



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Internal

Accessible only from the same assembly.

```
internal class RateCalculator  
{  
}
```

...

```
// In the same assembly  
var calc = new RateCalculator();
```

```
// In another assembly  
var calc = new RateCalculator();
```



Internal

Accessible only from the same assembly.

```
internal class RateCalculator  
{  
}
```

...

```
// In the same assembly  
var calc = new RateCalculator();
```

```
// In another assembly  
var calc = new RateCalculator();
```



Protected Internal

Accessible only from the same assembly or any derived classes.

```
public class Customer
{
    protected internal void Weir
    {
    }
}
```



Demo Access Modifiers



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

```
namespace AccessModifiers
{
    public class Customer
    {
        public int Id { get; set; }
        public string Name { get; set; }

        public void Promote()
        {

        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            var customer = new Customer();
            customer.
        }
    }
}
```

Equals

(object obj):bool

GetHashCode

Determines whether the specified **Object** is equal to the current **Object**.

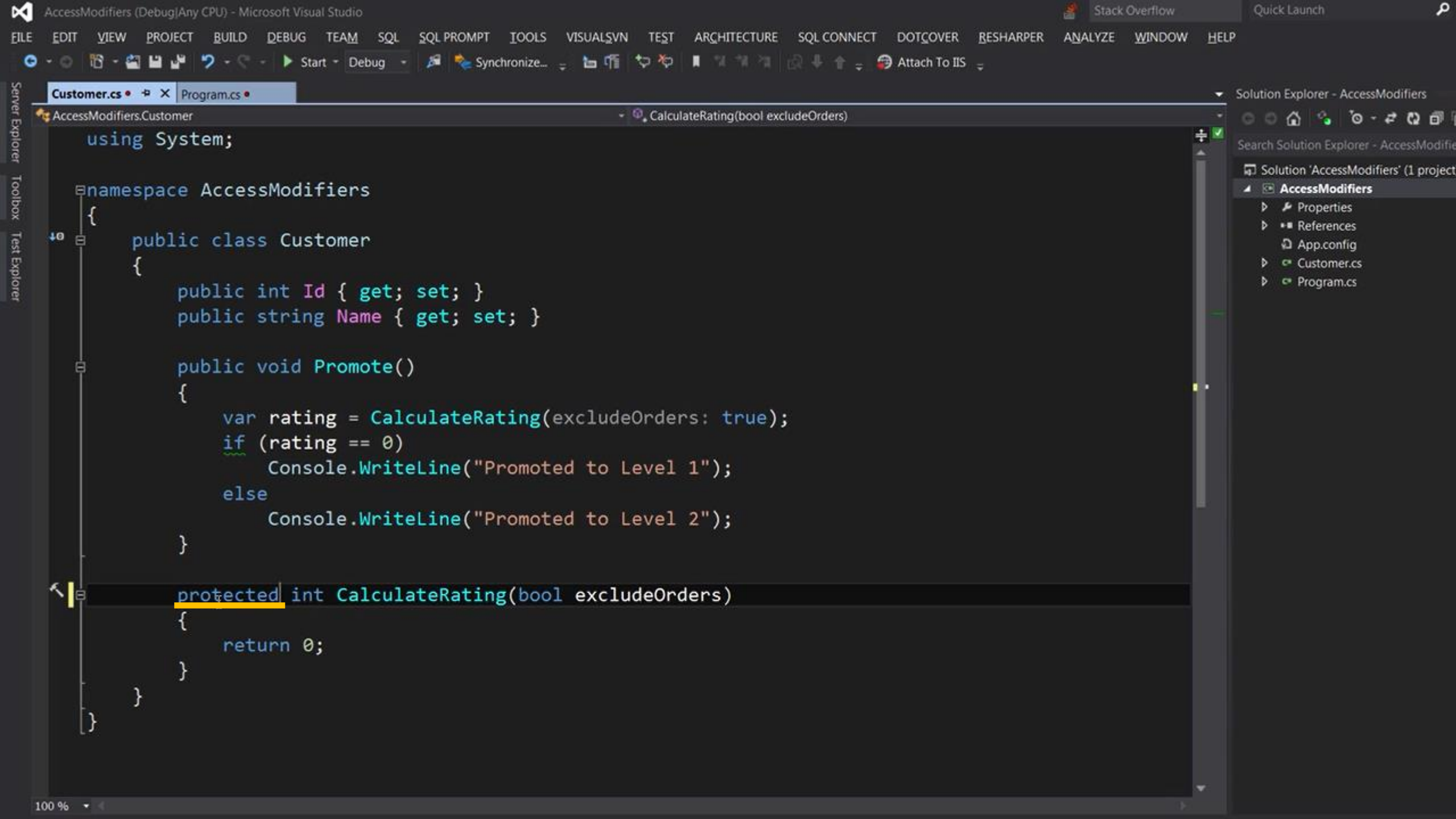
GetType

Id

Name

Promote

ToString





Customer.cs • Program.cs • X

AccessModifiers.GoldCustomer

OfferVouchar()

```
namespace AccessModifiers
{
    public class GoldCustomer : Customer
    {
        public void OfferVouchar()
        {
            this.CalculateRating(excludeOrders: true);
        }
    }

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

Solution Explorer - AccessModifiers

Search Solution Explorer - AccessModifiers

Solution 'AccessModifiers' (1 project)

AccessModifiers

- Properties
- References
- App.config
- Customer.cs
- Program.cs

Toolbox Test Explorer

AccessModifiers

- Properties
- References
 - Amazon
 - Microsoft.CSharp
 - System
 - System.Core
 - System.Data
 - System.Data.DataSetExtensions
 - System.Xml
 - System.Xml.Linq
- App.config
- Program.cs
- Amazon
 - Properties
 - References
 - Customer.cs

```
namespace Amazon
{
    public class Customer
    {
        public int Id { get; set; }
        public string Name { get; set; }

        public void Promote()
        {
            var calculator = new RateCalculator();
            var rating = calculator.Calculate(this);

            Console.WriteLine("Promote logic changed.");
        }
    }
}
```

Test Explorer

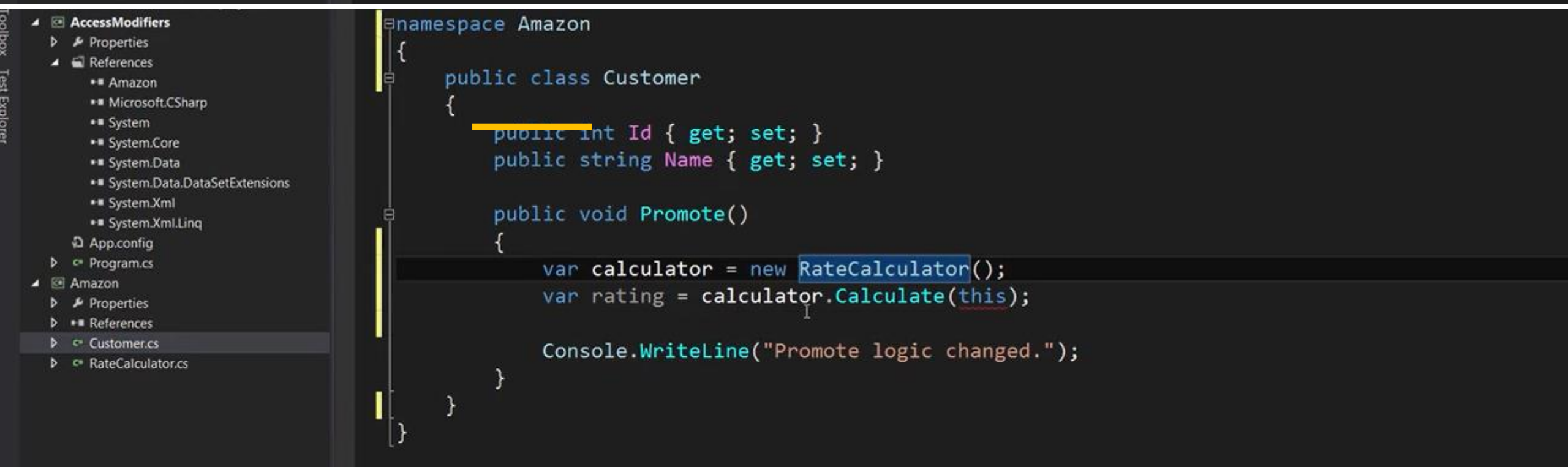
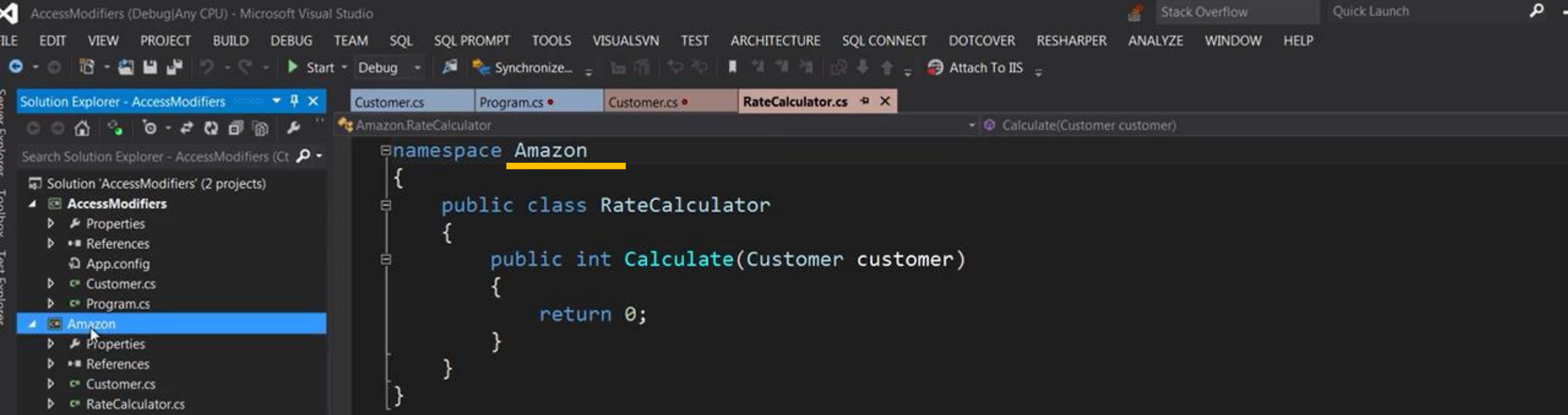
- Microsoft.CSharp
- System
- System.Core
- System.Data
- System.Data.DataSetExtensions
- System.Xml
- System.Xml.Linq
- App.config
- Program.cs
- Amazon
 - Properties
 - References
 - Customer.cs

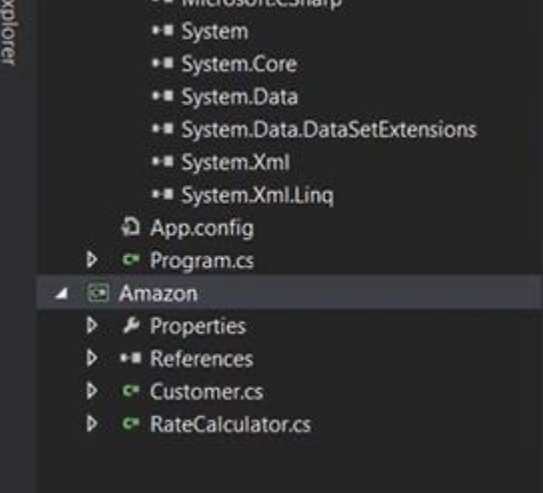
```
class Program
{
    static void Main(string[] args)
    {
        var customer = new Customer();
        Amazon.RateCalculator calculator = new RateCalculator();
    }
}
```



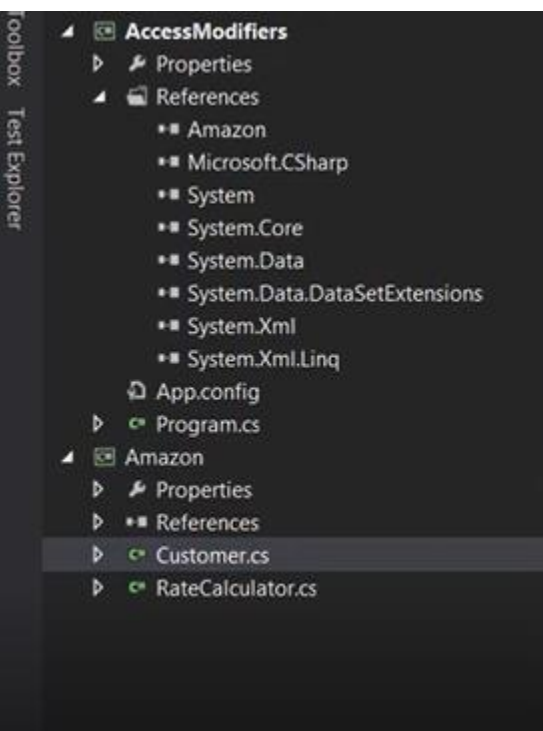
Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular





```
namespace Amazon
{
    internal class RateCalculator
    {
        public int Calculate()
        {
            return 0;
        }
    }
}
```



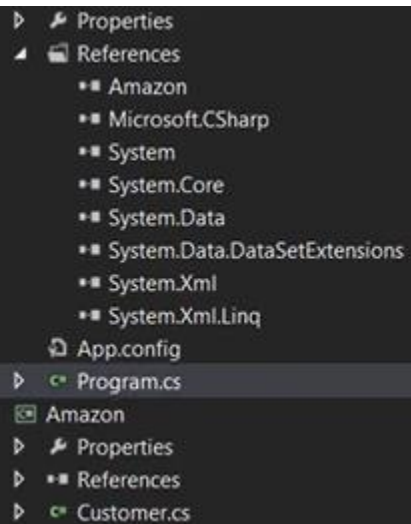
```
namespace Amazon
{
    public class Customer
    {
        public int Id { get; set; }
        public string Name { get; set; }

        public void Promote()
        {
            var calculator = new RateCalculator();
            var rating = calculator.Calculate(this);
            Console.WriteLine("Promote logic changed.");
        }
    }
}
```



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular



```
namespace AccessModifiers
{
    class Program
    {
        static void Main(string[] args)
        {
            var customer = new Customer();
            Amazon.RateCalculator calculator = new RateCalculator();
        }
    }
}
```



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Constructors and Inheritance



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Constructor Inheritance

- Base class constructors are always executed first.
- Base class constructors are not inherited.



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Constructor Inheritance

```
public class Vehicle
{
    private string _registrationNumber;

    public Vehicle(string registrationNumber)
    {
        _registrationNumber = registrationNumber;
    }
}
```



Constructor Inheritance

```
public class Car : Vehicle
{
    public Car(string registrationNumber)
    {
        _registrationNumber = registrationNumber;
    }
}
```



The base keyword

```
public class Car : Vehicle
{
    public Car(string registrationNumber)
        : base(registrationNumber)
    {
        // Initialise fields specific to the Car class
    }
}
```

4



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Demo Constructors and Inheritance



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

```
Test Explorer
40 public class Vehicle
    {
        public Vehicle()
        {
            Console.WriteLine("Vehicle is being initialized.");
        }
    }
}
```

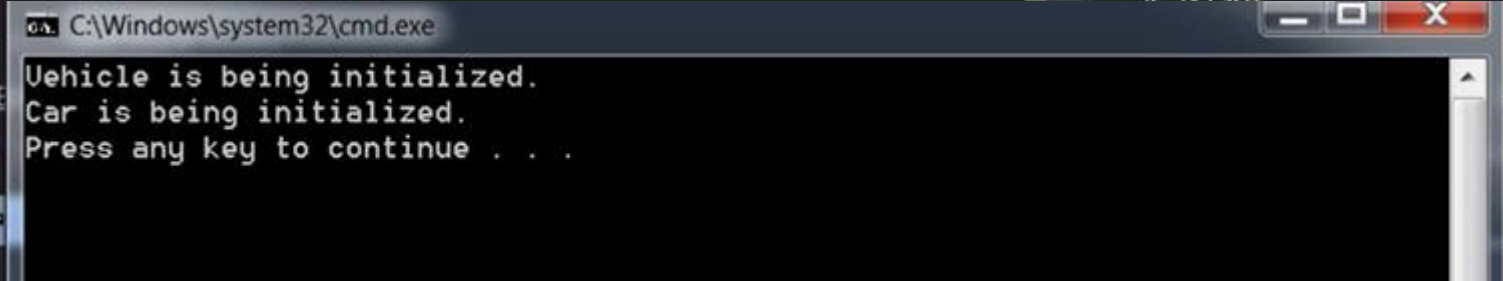
References
App.config
Program.cs
Vehicle.cs

```
Test Explorer
public class Car : Vehicle
{
    public Car()
    {
        Console.WriteLine("Car is being initialized.");
    }
}
```

References
App.config
Car.cs
Program.cs
Vehicle.cs

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

Properties
References
App.config
Car.cs



C:\Windows\system32\cmd.exe
Vehicle is being initialized.
Car is being initialized.
Press any key to continue . . .

```
public class Vehicle
{
    private readonly string _registrationNumber;

    //
    //
    //
    //
    public Vehicle()
    {
        Console.WriteLine("Vehicle is being initialized.");
    }

    public Vehicle(string registrationNumber)
    {
        _registrationNumber = registrationNumber;

        Console.WriteLine("Vehicle is being initialized. {0}", registrationNumber);
    }
}
```

- References
- App.config
- Car.cs
- Program.cs
- Vehicle.cs

```
public class Car : Vehicle
{
    public Car(string registrationNumber)
        : base(registrationNumber)
    {
        Console.WriteLine("Car is being initialized.");
    }
}
```

- References
- App.config
- Car.cs
- Program.cs
- Vehicle.cs



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Visual Studio interface showing the code editor and the command prompt output.

Code Editor (Program.cs):

```
namespace Constructors
{
    class Program
    {
        static void Main(string[] args)
        {
            var car = new Car("XYZ1234");
        }
    }
}
```

Command Prompt Output:

```
C:\Windows\system32\cmd.exe
Vehicle is being initialized. XYZ1234
Car is being initialized. XYZ1234
Press any key to continue . . .
```



Upcasting / Downcasting



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Agenda

- Conversion from a derived class to a base class (upcasting)
- Conversion from a base class to a derived class (downcasting)
- The as and is keywords



Upcasting

```
public class Shape  
{  
}
```

```
public class Circle : Shape  
{  
}
```



Upcasting

```
Circle circle = new Circle();  
Shape shape = circle;
```



Downcasting

```
Circle circle = new Circle();  
Shape shape = circle;
```

```
Circle anotherCircle = (Circle)shape;
```

```
Car car = (Car)shape; // throws InvalidCastException
```



The as keyword

```
Car car = (Car) obj;
```

```
Car car = obj as Car;  
if (car != null)  
{  
    ...  
}
```



The is keyword

```
if (obj is Car)
{
    Car car = (Car) obj;
    ...
}
```



Demo Upcasting / Downcasting



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

```
public class Shape
{
    public int Width { get; set; }
    public int Height { get; set; }
    public int X { get; set; }
    public int Y { get; set; }

    public void Draw()
    {

    }
}
```

Casting

- Properties
- References
- App.config
- Program.cs
- Shape.cs

```
public class Text : Shape
{
    public int FontSize { get; set; }
    public string FontName { get; set; }
}
```

Solution Casting (1 project)

Casting

- Properties
- References
- App.config
- Program.cs
- Shape.cs
- Text.cs

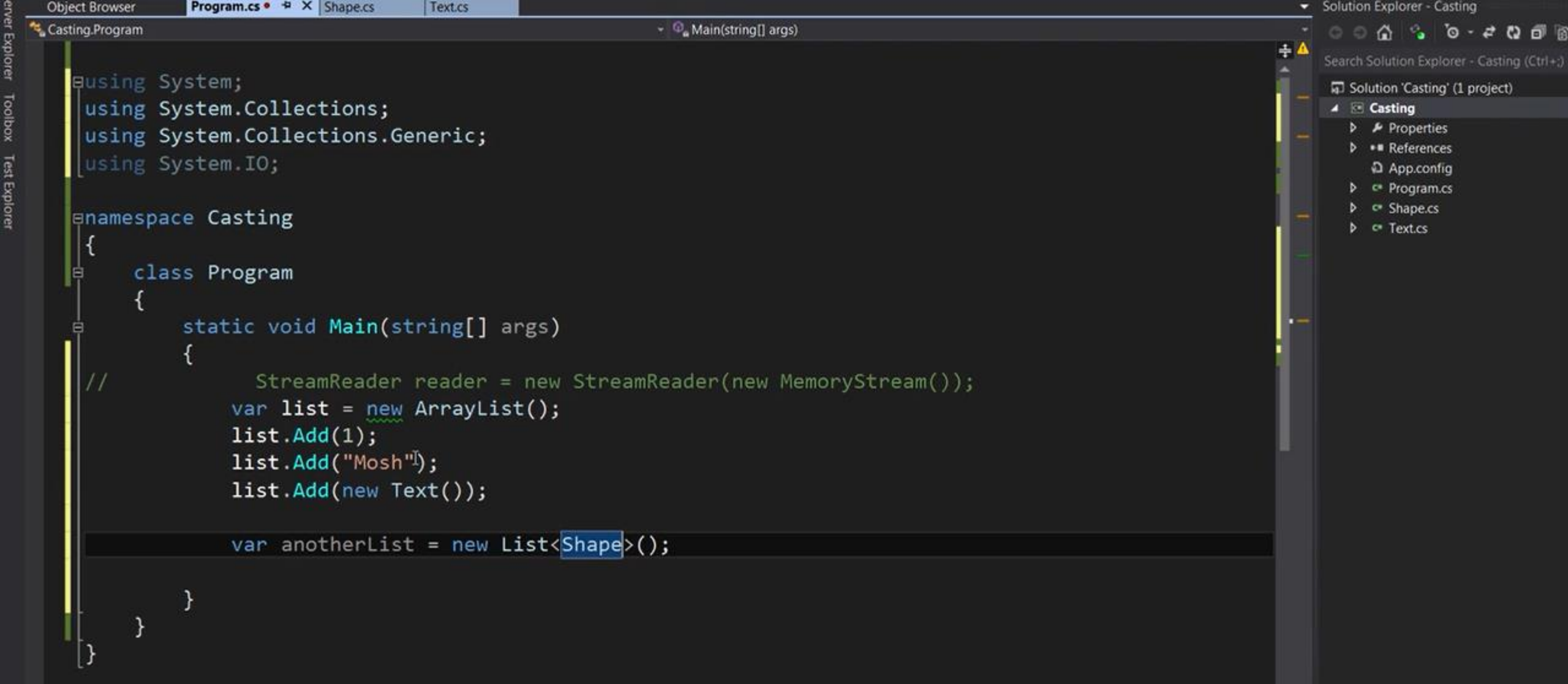
```
static void Main(string[] args)
{
    Text text = new Text();
    Shape shape = text;

    text.Width = 200;
    shape.Width = 100;

    Console.WriteLine(text.Width);
}
```

C:\Windows\system32\cmd.exe

100
Press any key to continue . . .



```
namespace Casting
{
    class Program
    {
        static void Main(string[] args)
        {
            Shape shape = new Text();
            Text text = (Text) shape;
            text.

        }
    }
}
```

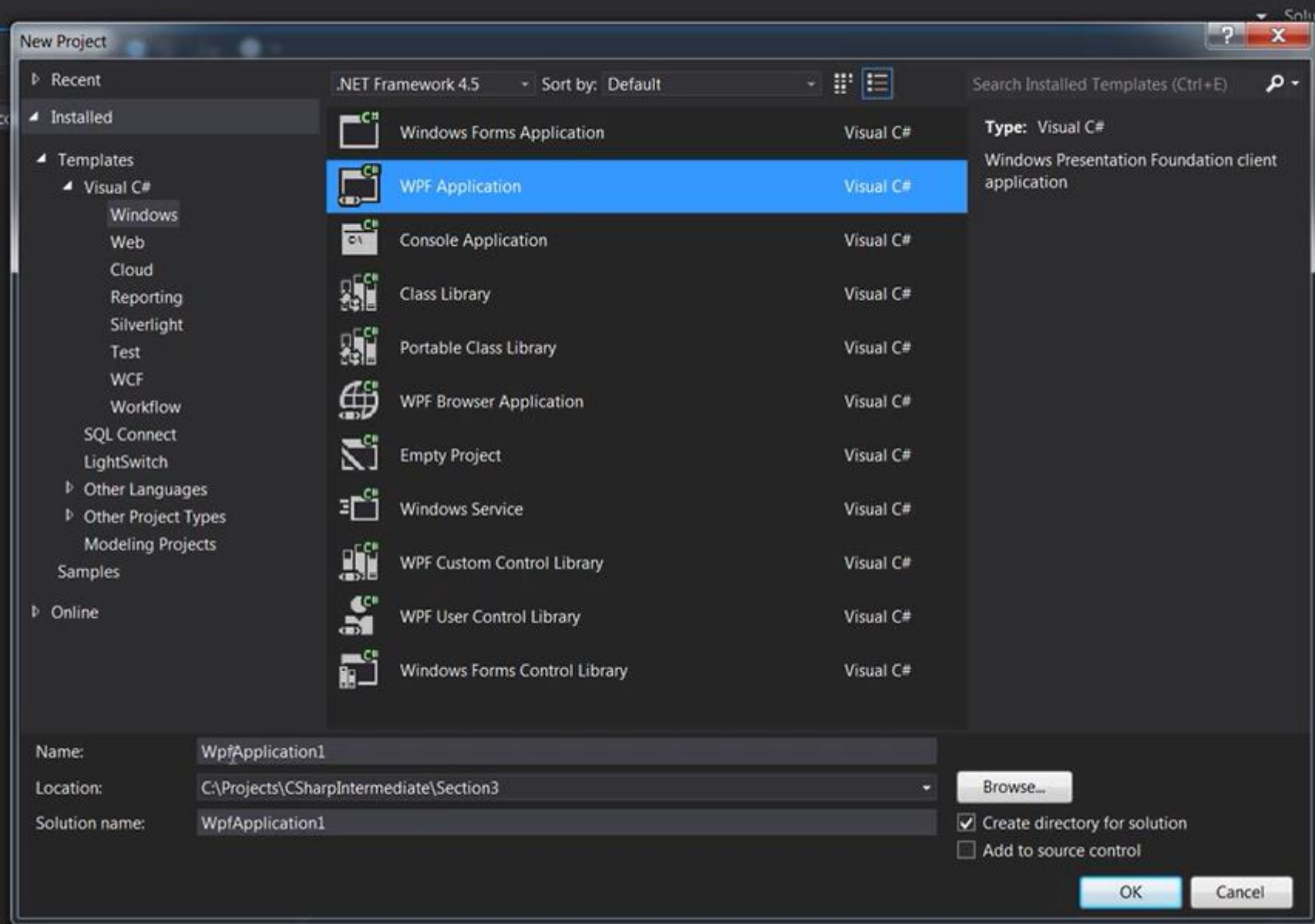
- Draw
- Equals
- FontName
- FontSize Property int Casting.Text.FontSize
- GetHashCode
- GetType
- Height
- ToString
- Width
- X
- Y

Shape.cs
Text.cs



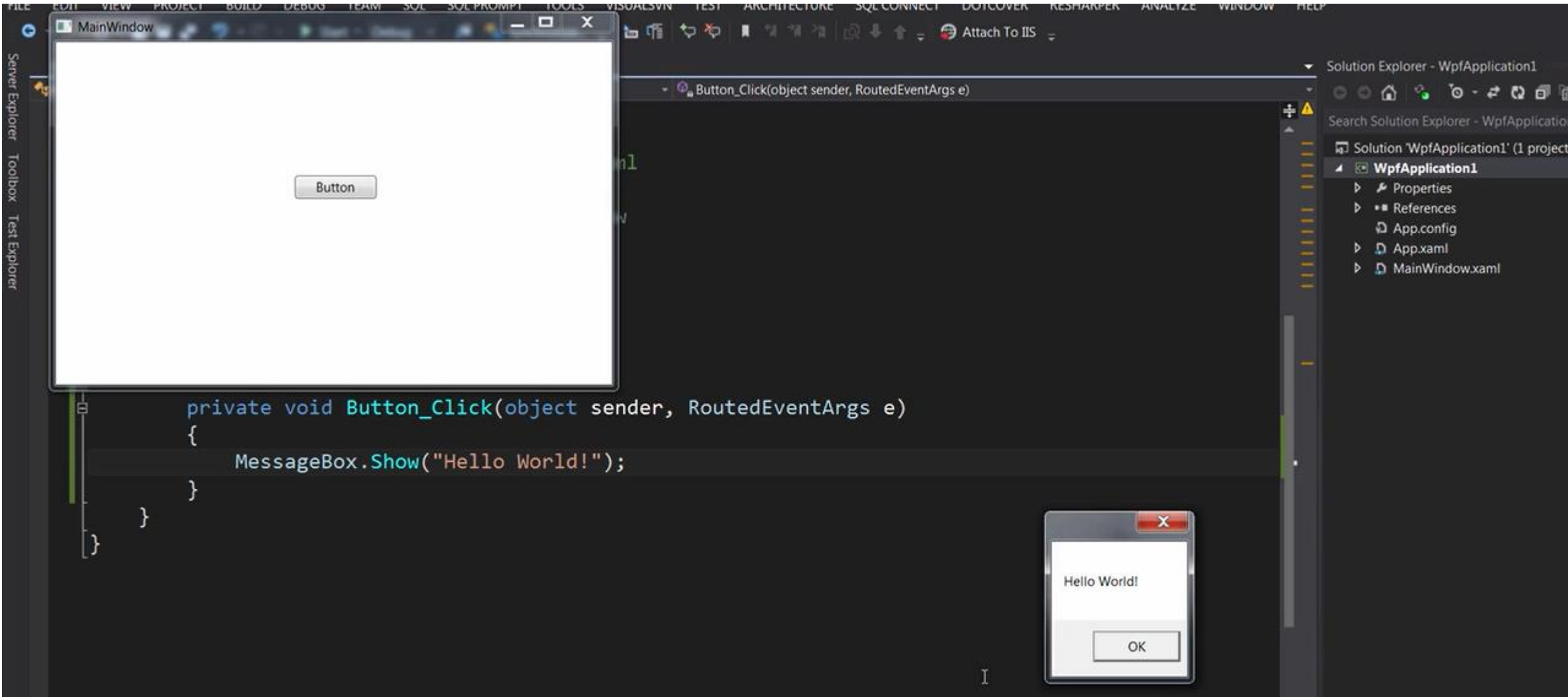
Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular



Object Browser MainWindow.xaml MainWindow.xaml.cs

WpfApplication1.MainWindow

```
/// <summary>
/// Interaction logic for MainWindow.xaml
/// </summary>
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        var button = (Button) sender;
        button.
```

Messa

- InputBindings
- InputHitTest
- InputScope
- InvalidateArrange
- InvalidateMeasure
- InvalidateProperty
- InvalidateVisual
- IsAncestorOf
- IsArrangeValid
- IsCancel
- IsDefault

Property double System.Windows.FrameworkElement.ActualHeight
Gets the rendered height of this element.

Solution Explorer - WpfApplication1

Search Solution Explorer - WpfApplication1

Solution 'WpfApplication1' (1 project)

- WpfApplication1
 - Properties
 - References
 - App.config
 - App.xaml
 - MainWindow.xaml



MainWindow

Button

21.96

OK

Button_Click(object sender, RoutedEventArgs e)

```
{
    var button = sender as Button;
    if (button != null)
    {
        MessageBox.Show(button.ActualHeight.ToString());
    }

    MessageBox.Show("Hello World!");
}
```

Solution Explorer - WpfApplication1

Search Solution Explorer - WpfApplication1

Solution 'WpfApplication1' (1 project)

- WpfApplication1
 - Properties
 - References
 - App.config
 - App.xaml
 - MainWindow.xaml



Boxing and Unboxing



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Agenda

- Value Types and Reference Types
- Boxing and Unboxing



Types in C#

- Value types
- Reference types



Value Types

- Are stored on the stack.
- Examples:
 - All primitive types: byte, int, float, char, bool
 - The struct type



Reference Types

- Are stored in the heap.
- Examples:
 - Any classes (Object, Array, String, DbMigrator, etc)



Earlier in this section...

- An object reference can be implicitly converted to a base class reference.

```
Circle circle = new Circle();  
Shape shape = circle;
```



Also...

- The Object class is the base of all classes in .NET Framework.

```
Circle circle = new Circle();  
Shape shape = circle;  
object shape = circle;
```



Boxing

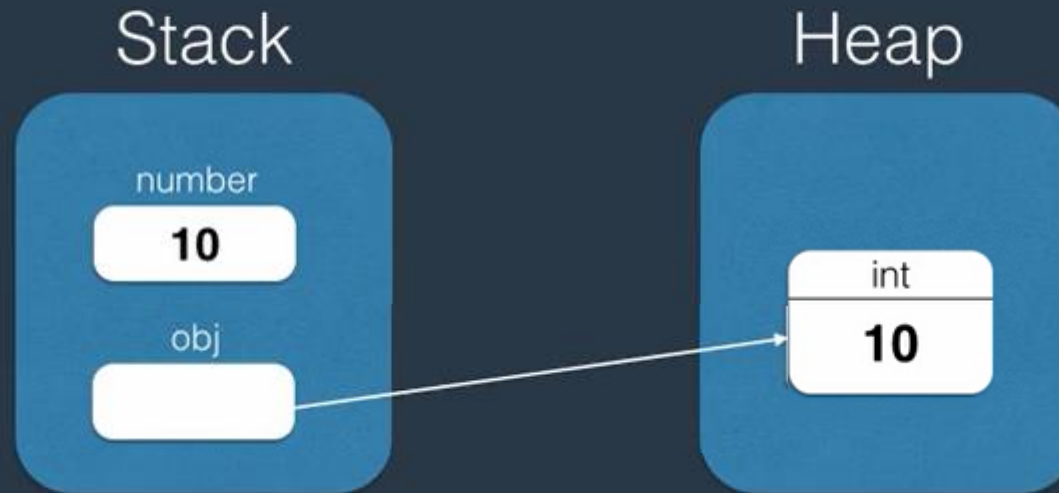
- The process of converting a value type instance to an object reference

```
int number = 10;  
object obj = number;
```

```
// or  
object obj = 10;
```



Boxing



4



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Unboxing

```
object obj = 10;  
int number = (int)obj;
```



Boxing / Unboxing

- Have a performance penalty.



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Demo Boxing and Unboxing



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Boxing (Debug)Any CPU - Microsoft Visual Studio

Stack Overflow Quick Launch

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL SQL PROMPT TOOLS VISUALSVN TEST ARCHITECTURE SQL CONNECT DOTCOVER RESHARPER ANALYZE WINDOW HELP

Start Debug Synchronize Attach To IIS

Program.cs

Boxing.Program

Main(string[] args)

```
using System;
using System.Collections;

namespace Boxing
{
    class Program
    {
        static void Main(string[] args)
        {
            var list = new ArrayList();
            list.Add(1);
            list.Add("Mosh");
            list.Add(DateTime.Today);

            var number = (int) list[1];
        }
    }
}
```

Solution Explorer - Boxing

Search Solution Explorer - Boxing (Ctrl+;)

Solution 'Boxing' (1 project)

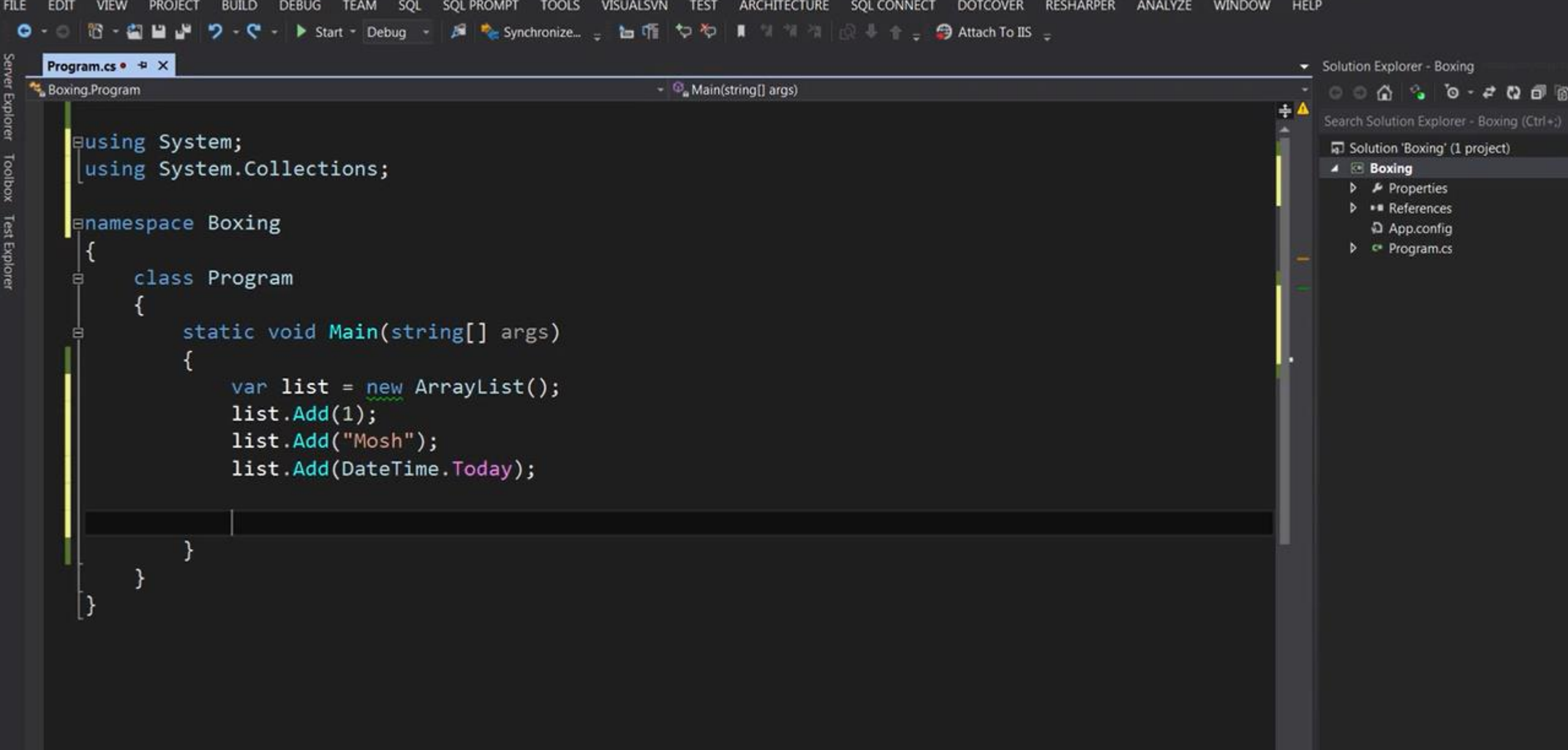
- Boxing
 - Properties
 - References
 - App.config
 - Program.cs

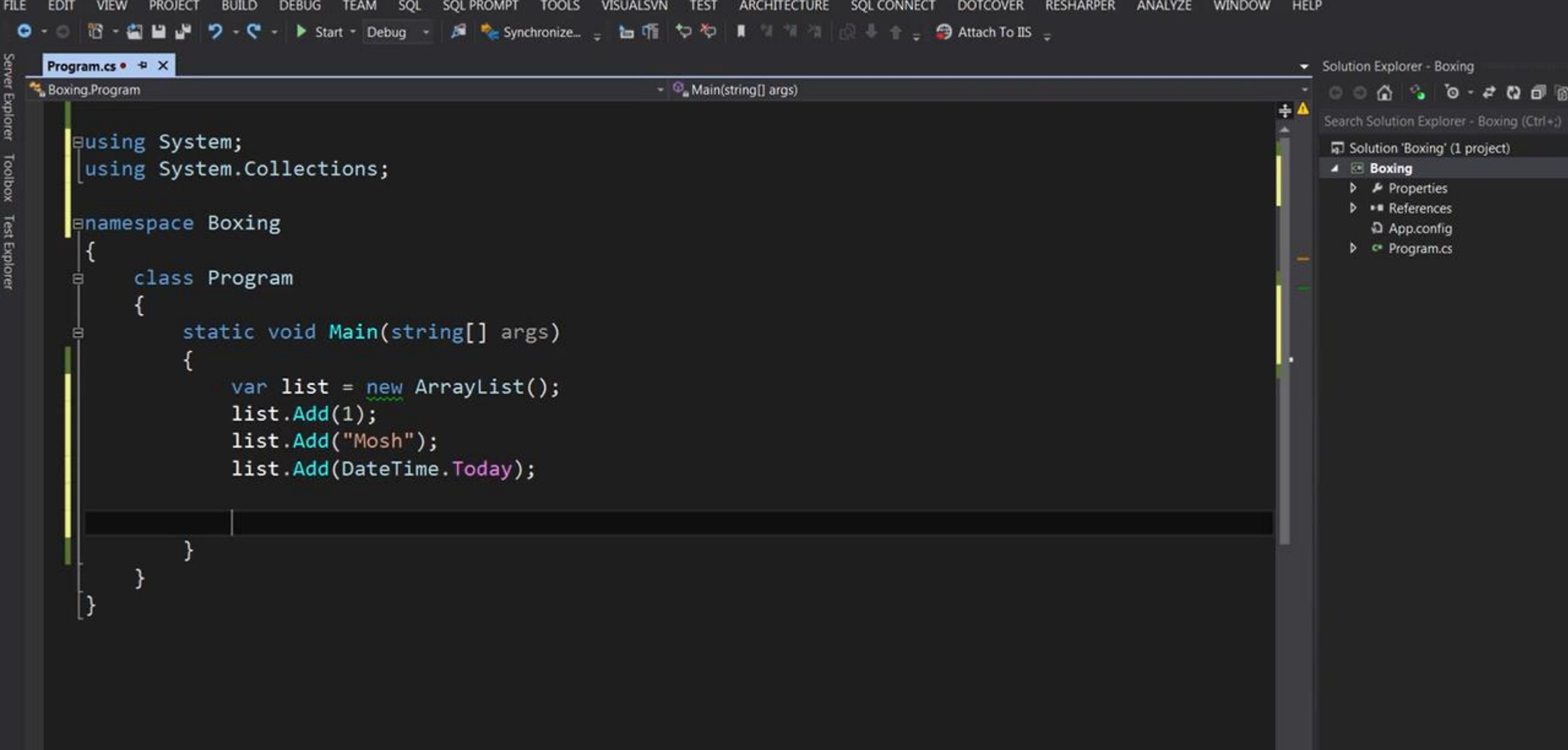
C# Intermediate



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular





INHERITANCE - Exercises



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Exercise: Design a Stack

A Stack is a data structure for storing a list of elements in a LIFO (last in, first out) fashion. Design a class called Stack with three methods.

```
void Push(object obj)
object Pop()
void Clear()
```

The **Push()** method stores the given object on top of the stack. We use the “object” type here so we can store any objects inside the stack. Remember the “object” class is the base of all classes in the .NET Framework. So any types can be automatically upcast to the object. Make sure to take into account the scenario that null is passed to this object. We should not store null references in the stack. So if null is passed to this method, you should throw an `InvalidOperationException`. Remember, when coding every method, you should think of all possibilities and make sure the method behaves properly in all these edge cases. That’s what distinguishes you from an “average” programmer.

The **Pop()** method removes the object on top of the stack and returns it. Make sure to take into account the scenario that we call the `Pop()` method on an empty stack. In this case, this method should throw an `InvalidOperationException`. Remember, your classes should always be in a valid state and used properly. When they are misused, they should throw exceptions. Again, thinking of all these edge cases, separates you from an average programmer. The code written this way will be more robust and with less bugs.

The **Clear()** method removes all objects from the stack.

We should be able to use this stack class as follows:

```
var stack = new Stack();
stack.Push(1);
stack.Push(2);
stack.Push(3);
Console.WriteLine(stack.Pop());
Console.WriteLine(stack.Pop());
Console.WriteLine(stack.Pop());
```

The output of this program will be

```
3
2
1
```