C# INTERMEDIATE

Methods

Agenda

- Signature of Methods
- Method Overloading
- Params modifier
- Ref modifier
- Out modifier

Signature of a Method

- Name
- Number and Type of parameters

```
public class Point
{
    public void Move(int x, int y) {}
}
```

Overloading Methods

- Having a method with the same name but different signatures

```
public class Point
{
    public void Move(int x, int y) {}

    public void Move(Point newLocation) {}

    public void Move(Point newLocation, int speed) {}
}
```

A method with varying number of parameters

```
public class Calculator
{
    public int Add(int n1, int n2){}
    public int Add(int n1, int n2, int n3){}
    public int Add(int n1, int n2, int n3, int n4){}
    …
}
```

A method with varying number of parameters

```
public class Calculator
{
    public int Add(int[] numbers){}
}


var result = calculator.Add(new int[]{ 1, 2, 3, 4 });
```

The Params Modifier

```
public class Calculator
{
    public int Add(params int[] numbers){}
}


var result = calculator.Add(new int[]{ 1, 2, 3, 4 });
var result = calculator.Add(1, 2, 3, 4);
```

The Ref Modifier

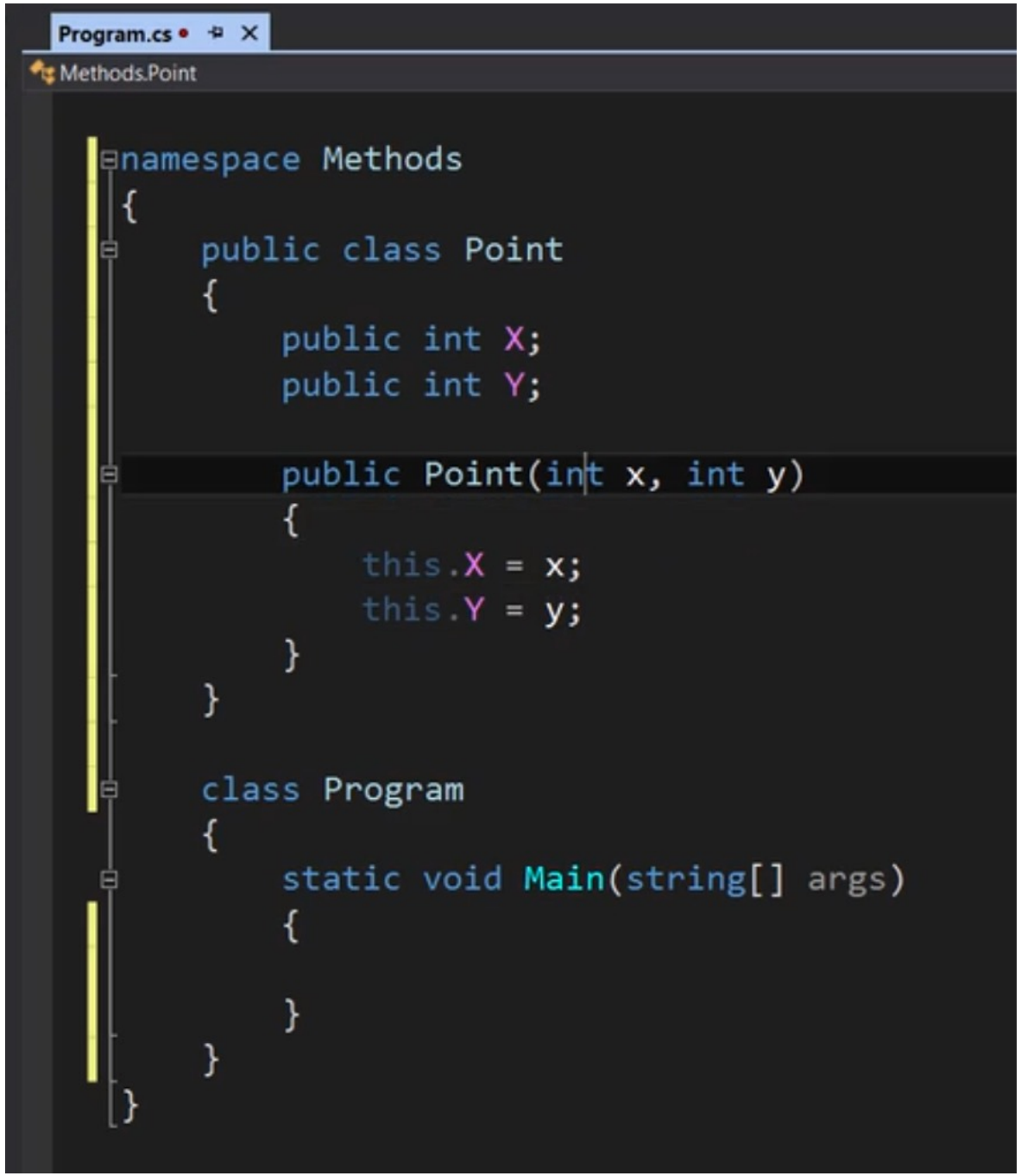
```
public class MyClass
{
    public void MyMethod(int a)
    {
        a += 2;
    }
}

var a = 1;
myClass.MyMethod(a);
```

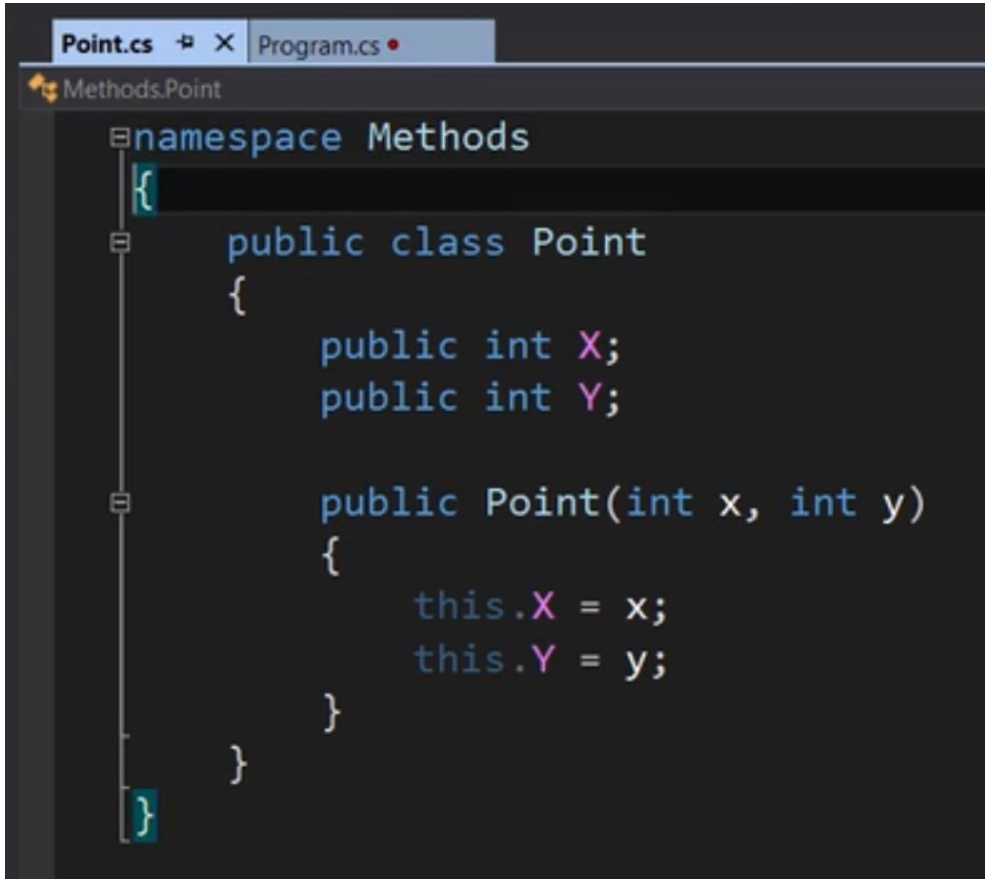
The Ref Modifier

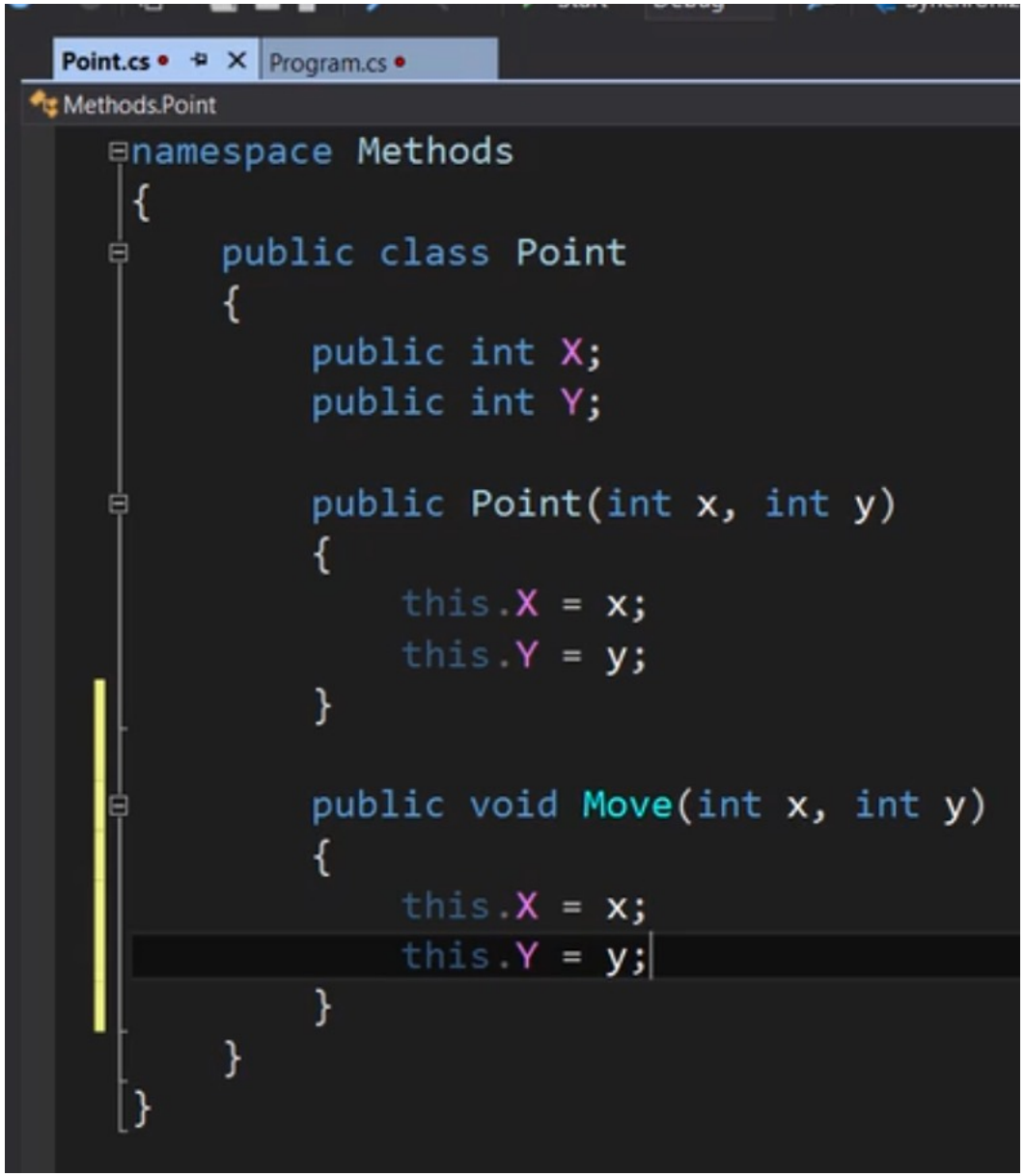
```
public class Weirdo
{
    public void DoAWeirdThing(ref int a)
    {
        a += 2;
    }
}

var a = 1;
weirdo.DoAWeirdThing(ref a);
```

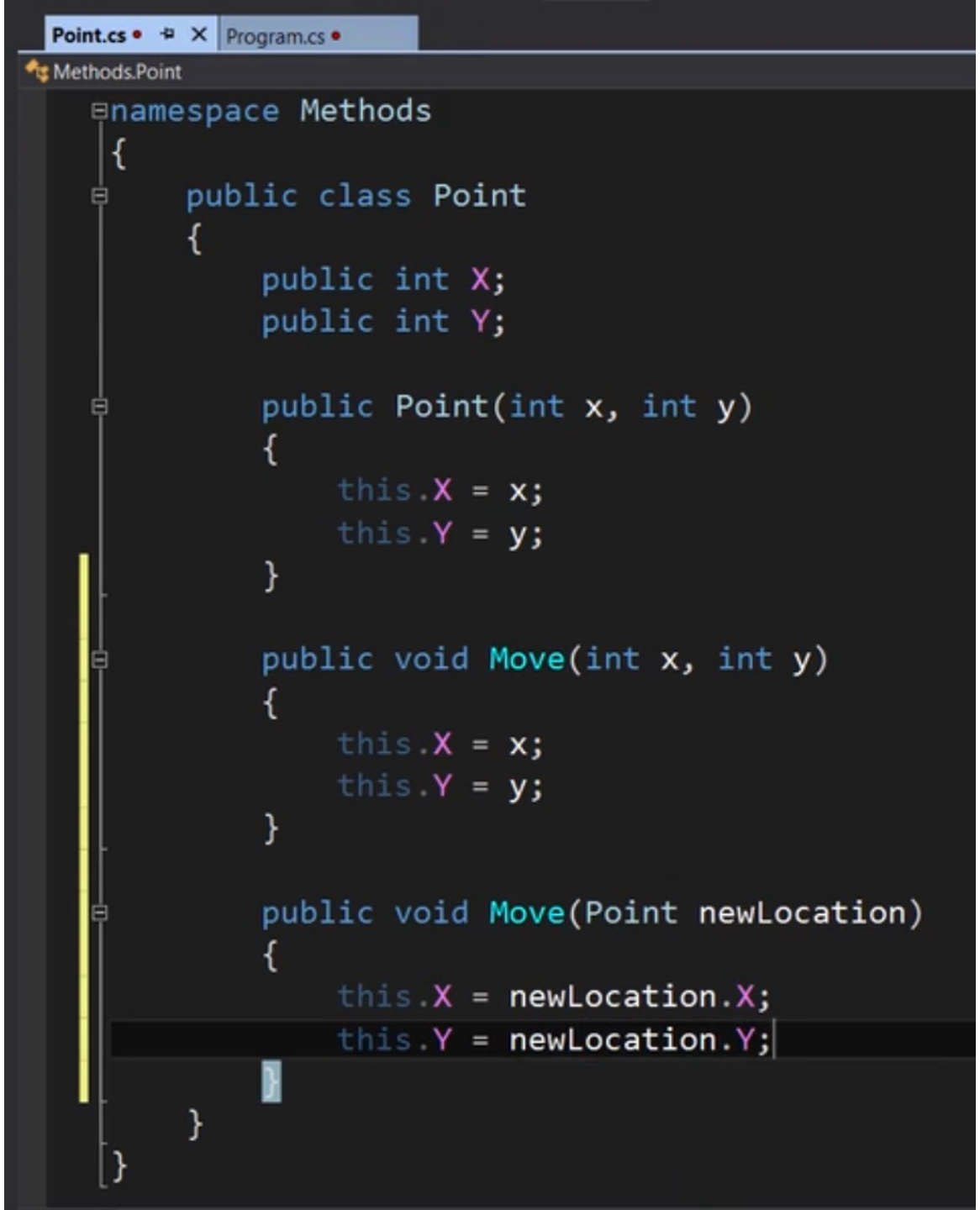
The Out Modifier

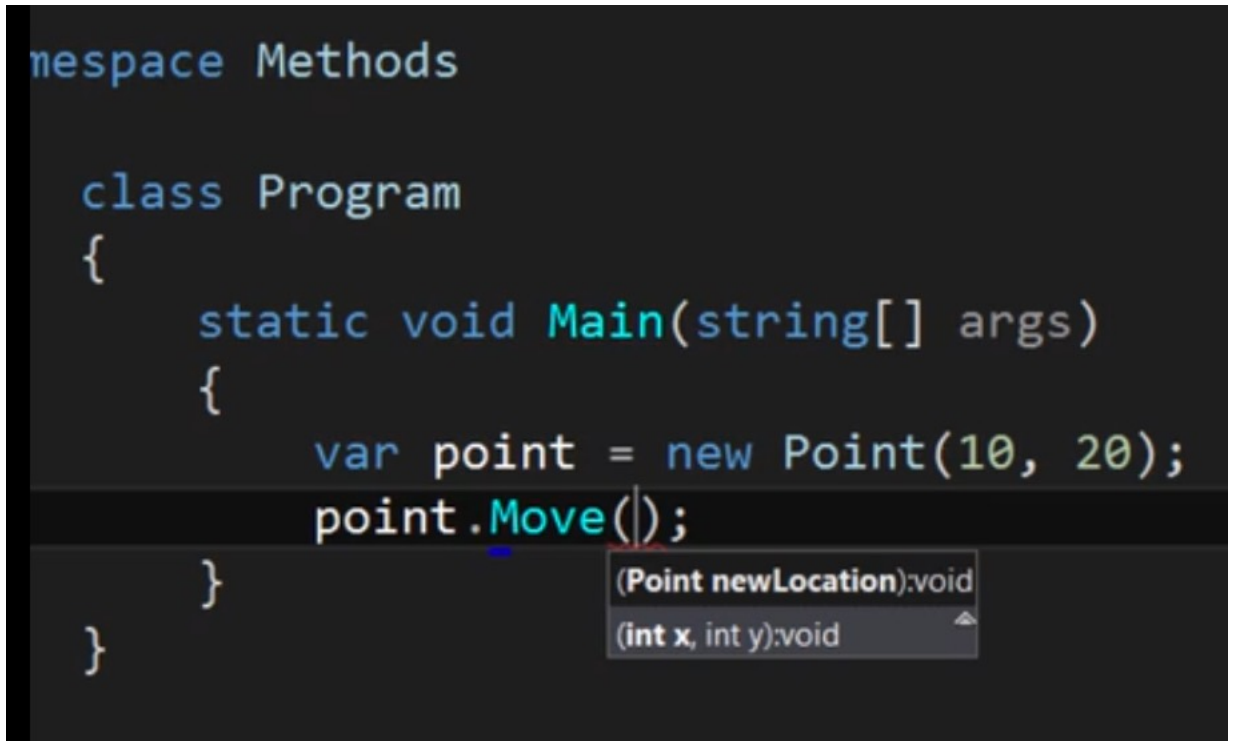
```
public class MyClass
{
    public void MyMethod(out int result)
    {
        result = 1;
    }
}

int a;
myClass.MyMethod(out a);
```

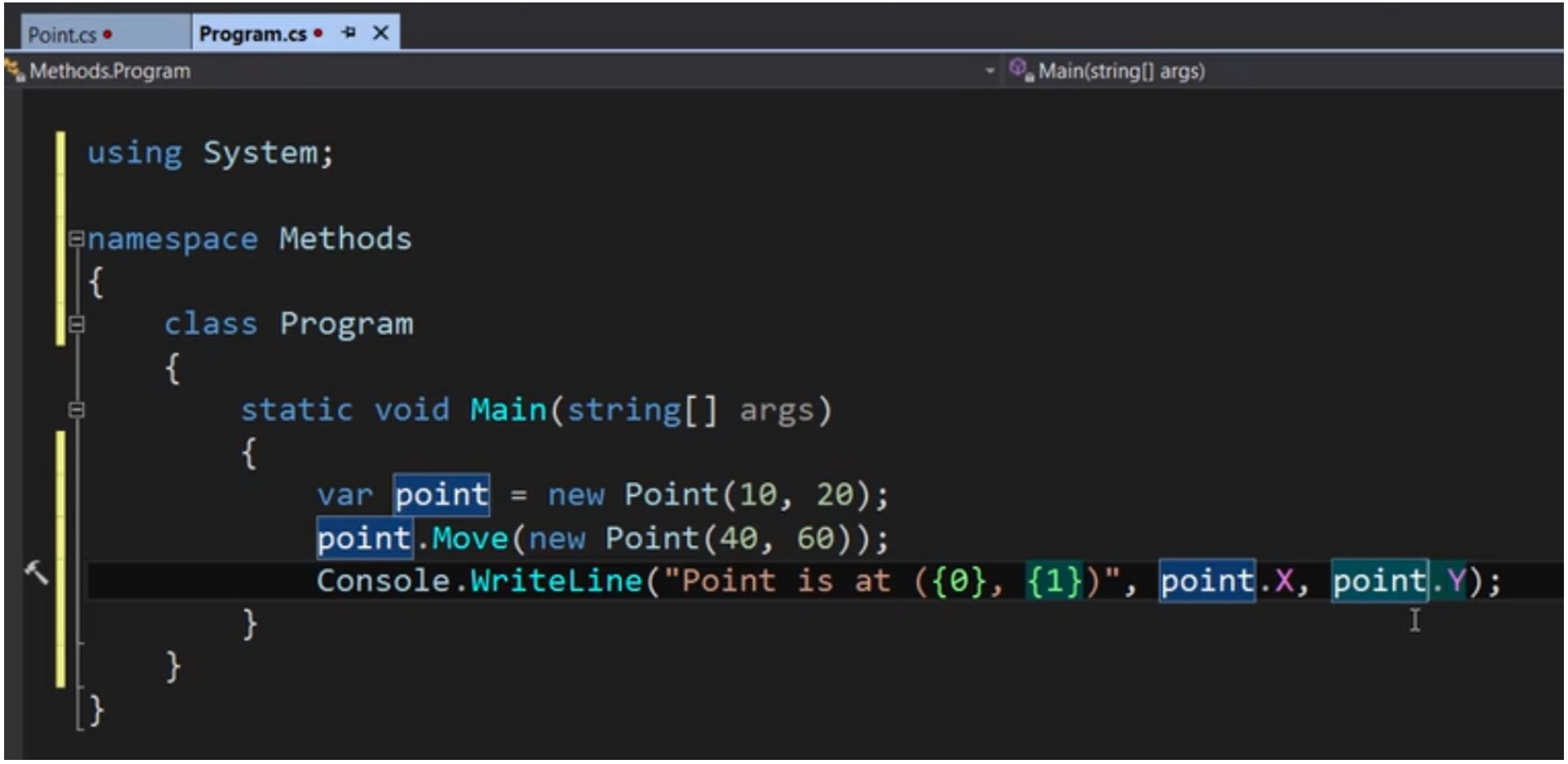

Program.cs
Methods.Point
namespace Methods
{
public class Point
{
public int X;
public int Y;
public Point(int x, int y)
{
this.X = x;
this.Y = y;
}
}
class Program
{
static void Main(string[] args)
{
}
}
}

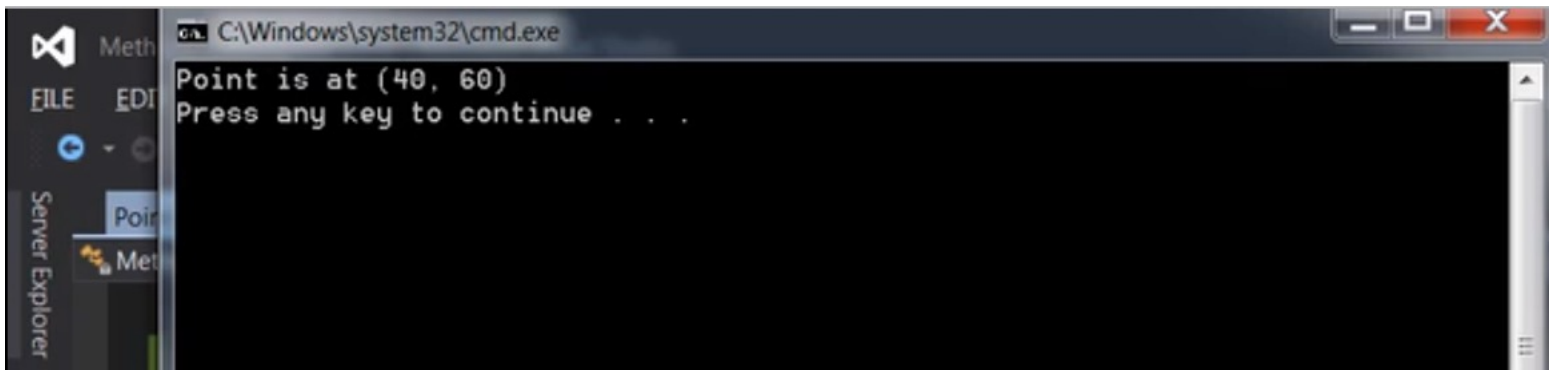

Point.cs
Program.cs
Methods.Point
namespace Methods
{
public class Point
{
public int X;
public int Y;
public Point(int x, int y)
{
this.X = x;
this.Y = y;
}
}
}

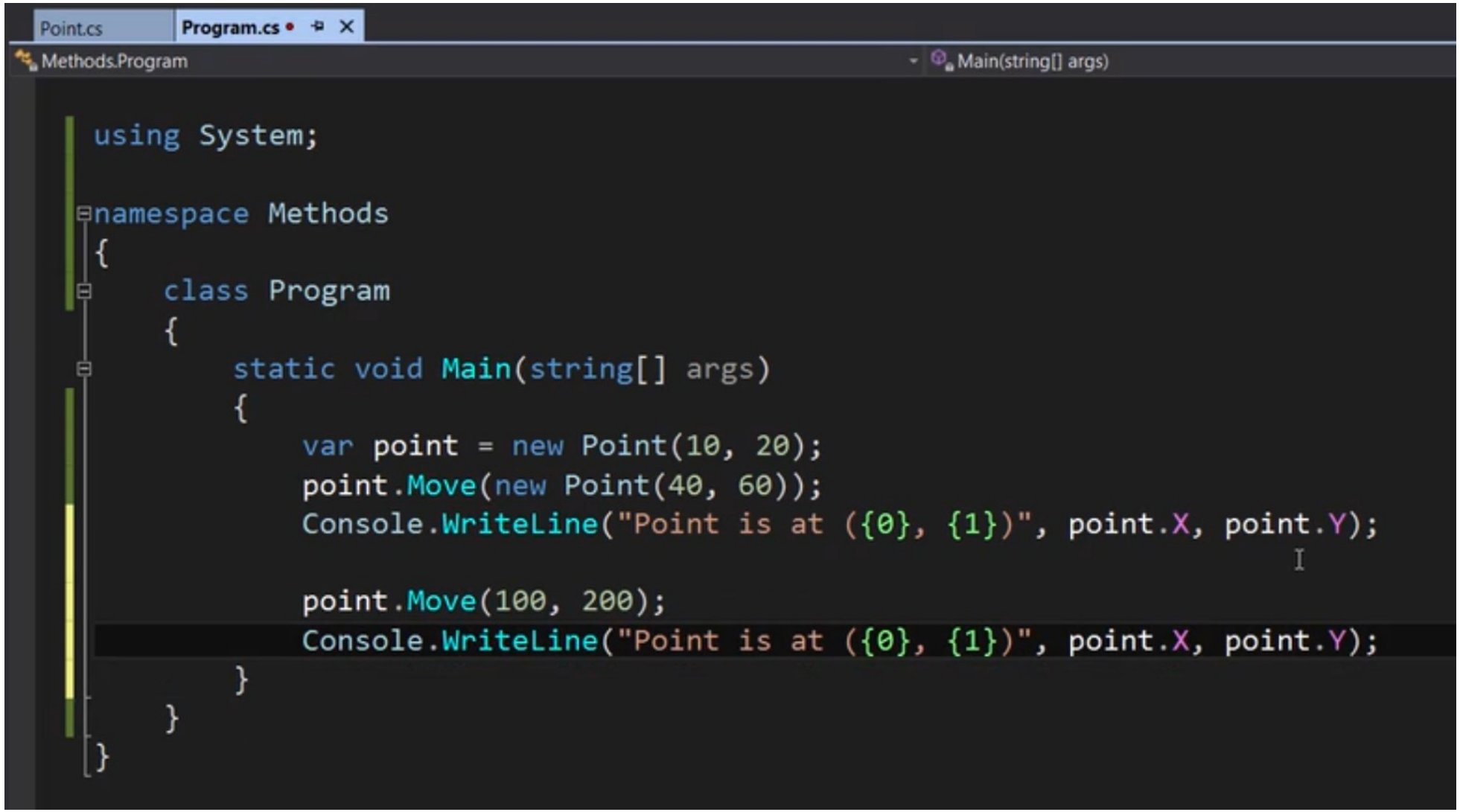

Point.cs
Program.cs
Methods.Point
namespace Methods
{
public class Point
{
public int X;
public int Y;
public Point(int x, int y)
{
this.X = x;
this.Y = y;
}
public void Move(int x, int y)
{
this.X = x;
this.Y = y;
}
}
}

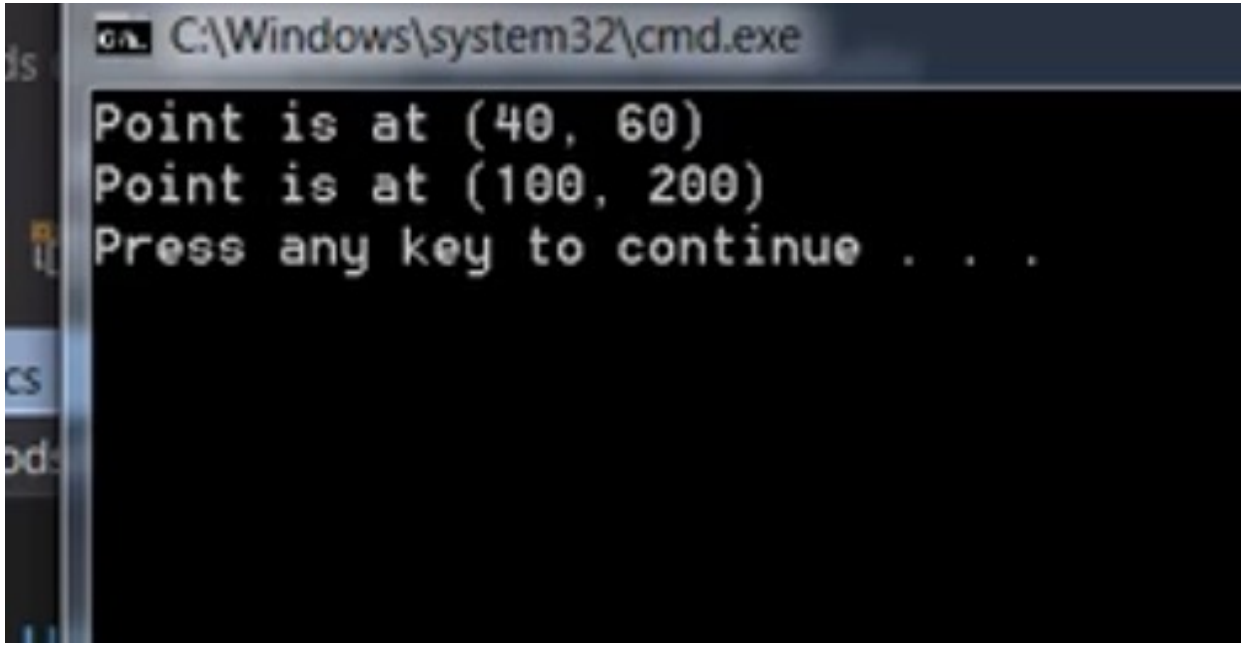

Point.cs
Program.cs
Methods.Point
namespace Methods
{
public class Point
{
public int X;
public int Y;
public Point(int x, int y)
{
this.X = x;
this.Y = y;
}
public void Move(int x, int y)
{
this.X = x;
this.Y = y;
}
public void Move(Point newLocation)
{
this.X = newLocation.X;
this.Y = newLocation.Y;
}
}
}


mespace Methods
class Program
{
static void Main(string[] args)
{
var point = new Point(10, 20);
point.Move();
}
}
(Point newLocation):void
(int x, int y):void


Point.cs
Program.cs
Methods.Program
Main(string[] args)
using System;
namespace Methods
{
class Program
{
static void Main(string[] args)
{
var point = new Point(10, 20);
point.Move(new Point(40, 60));
Console.WriteLine("Point is at ({0}, {1})", point.X, point.Y);
}
}
}


Meth
FILE
EDI
Server Explorer
Poir
Met
C:\Windows\system32\cmd.exe
Point is at (40, 60)
Press any key to continue . . .


Point.cs
Program.cs
Methods.Program
Main(string[] args)
using System;
namespace Methods
{
class Program
{
static void Main(string[] args)
{
var point = new Point(10, 20);
point.Move(new Point(40, 60));
Console.WriteLine("Point is at ({0}, {1})", point.X, point.Y);
point.Move(100, 200);
Console.WriteLine("Point is at ({0}, {1})", point.X, point.Y);
}
}
}


C:\Windows\system32\cmd.exe
Point is at (40, 60)
Point is at (100, 200)
Press any key to continue . . .