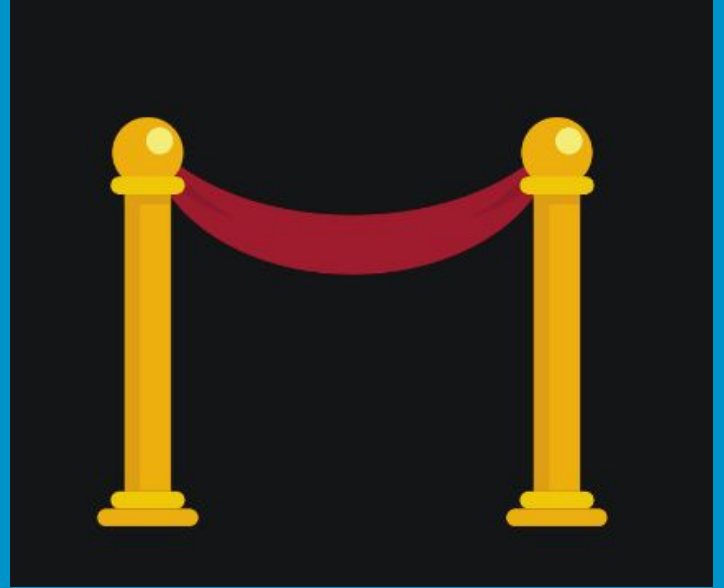


Encapsulation



OOPs A PIE

Stands for:

A - abstraction

P - polymorphism

I - inheritance

E - encapsulation



Access

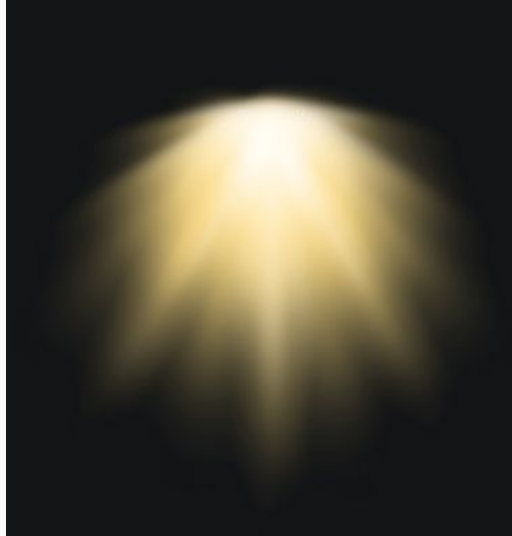


PARTY!!!

The image features a central text element 'PARTY!!!' rendered in a bright, glowing neon font with a blue-to-white gradient. The text is enclosed within a rounded rectangular frame that also has a glowing neon border. The background is solid black, which makes the colorful elements stand out. Scattered around the central text are various party-related decorations: long, flowing streamers in orange and yellow, and small, multi-colored confetti pieces in yellow, orange, blue, and pink. The overall composition is festive and celebratory.

Access Modifier

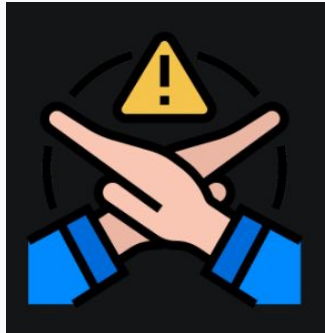
- **Access Modifier**: Defines the scope and visibility of a type or member



What's the point?

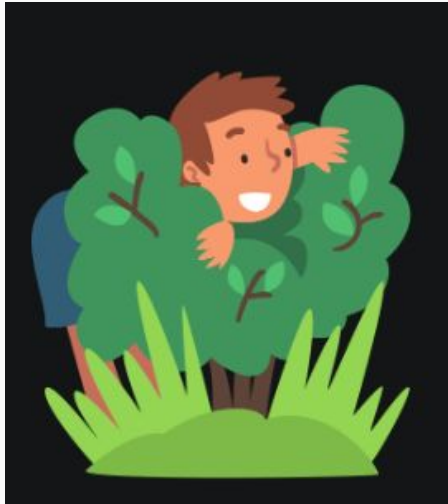
Encapsulation is used to restrict access to members of a class so as to prevent the user of a given class from manipulating objects in ways that are not intended by the designer.

- Encapsulation hides the internal implementation of functionalities of a class without affecting the overall functioning of the system.



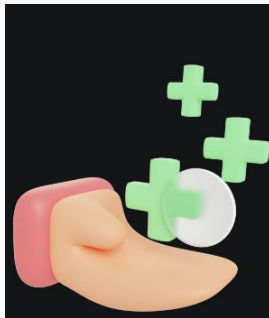
Information Hiding

- Note: Encapsulation is also called information hiding.



Encapsulation Benefits

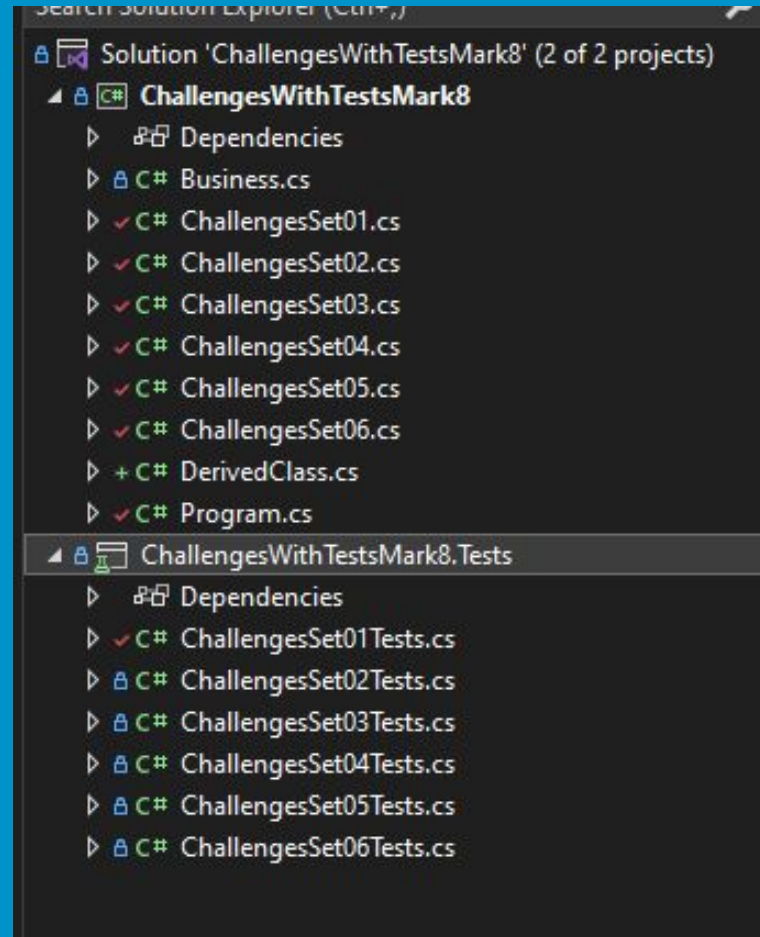
- C# provides **code security, flexibility, and easy maintainability** through encapsulation.
- Encapsulation is also useful in hiding the data of a class from illegal direct access.



5 Kinds of Access Modifiers

1. **public**
2. **private**
3. **protected**
4. **internal**
5. **protected internal**

Weekly Challenges Example



Public

- public - access modifier makes the member accessible from outside of the class

Public – Can access anywhere!

```
namespace ChallengesWithTestsMark8
```

```
{  
    public class ChallengesSet01
```

```
{  
    3 references  
    protected bool AreTwoNumbersTheSame(int num1, int num2)
```

```
{  
        if (num1 == num2)  
        {  
            return true;  
        }  
        else  
        {  
            return false;  
        }  
    }
```

```
    1 reference  
    public double Subtract(double minuend, double subtrahend)  
    {  
        AreTwoNumbersTheSame(5, 5);  
        return 2.2;  
    }
```

```
    1 reference  
    public int Add(int number1, int number2)  
    {  
        throw new NotImplementedException();  
    }
```

Solution 'ChallengesWithTestsMark8' (2 of 2 projects)

- ChallengesWithTestsMark8
 - Dependencies
 - Business.cs
 - ChallengesSet01.cs
 - ChallengesSet02.cs
 - ChallengesSet03.cs
 - ChallengesSet04.cs
 - ChallengesSet05.cs
 - ChallengesSet06.cs
 - DerivedClass.cs
 - Program.cs
- ChallengesWithTestsMark8.Tests
 - Dependencies
 - ChallengesSet01Tests.cs
 - ChallengesSet02Tests.cs
 - ChallengesSet03Tests.cs
 - ChallengesSet04Tests.cs
 - ChallengesSet05Tests.cs
 - ChallengesSet06Tests.cs

Private

- private - makes members accessible only from within the class it was created in and hides the member from the outside



Private – Class only!

```
1 using System;
2
3 namespace ChallengesWithTestsMark8
4 {
5     16 references
6     public class ChallengesSet01
7     {
8         2 references
9         private bool AreTwoNumbersTheSame(int num1, int num2)
10         {
11             if (num1 == num2)
12             {
13                 return true;
14             }
15             else
16             {
17                 return false;
18             }
19         }
20
21         1 reference
22         public double Subtract(double minuend, double subtrahend)
23         {
24             AreTwoNumbersTheSame(5, 5);
25             return 2.2;
26         }
27
28         1 reference
29         public int Add(int number1, int number2)
30         {
31             throw new NotImplementedException();
32         }
33     }
34 }
```

Search Solution Explorer (Ctrl+)

Solution 'ChallengesWithTestsMark8' (2 of 2 projects)

- ChallengesWithTestsMark8
 - Dependencies
 - Business.cs
 - ChallengesSet01.cs
 - ChallengesSet02.cs
 - ChallengesSet03.cs
 - ChallengesSet04.cs
 - ChallengesSet05.cs
 - ChallengesSet06.cs
 - DerivedClass.cs
 - Program.cs
- ChallengesWithTestsMark8.Tests
 - Dependencies
 - ChallengesSet01Tests.cs
 - ChallengesSet02Tests.cs
 - ChallengesSet03Tests.cs
 - ChallengesSet04Tests.cs
 - ChallengesSet05Tests.cs
 - ChallengesSet06Tests.cs

Private – Class only!

```
4 namespace ChallengesWithTestsMark8.Tests
5 {
6     0 references
7     public class ChallengesSet01Tests
8     {
9         [Theory]
10        [InlineData(1, 1, true)]
11        [InlineData(10, 10, true)]
12        [InlineData(99, 99, true)]
13        [InlineData(-10, -10, true)]
14        [InlineData(-1, -1, true)]
15        [InlineData(0, 1, false)]
16        [InlineData(4, 7, false)]
17        [InlineData(-1, 1, false)]
18        [InlineData(5, 6, false)]
19
20        0 references
21        public void AreTwoNumbersTheSameTest(int number1, int number2, bool expected)
22        {
23            // Arrange
24            ChallengesSet01 challenger = new ChallengesSet01();
25
26            // Act
27            bool actual = challenger.AreTwoNumbersTheSame(number1, number2);
28
29            // Assert
30            Assert.Equal(expected, actual);
31        }
32
33        [Theory]
34        [InlineData(10, 7, 3)]
35        [InlineData(100, 75, 25)]
36        [InlineData(1, 1, 0)]
37    }
```

Error

CS0122: 'ChallengesSet01.AreTwoNumbersTheSame(int, int)' is inaccessible due to its protection level

Show potential fixes (Alt+Enter or Ctrl+.)

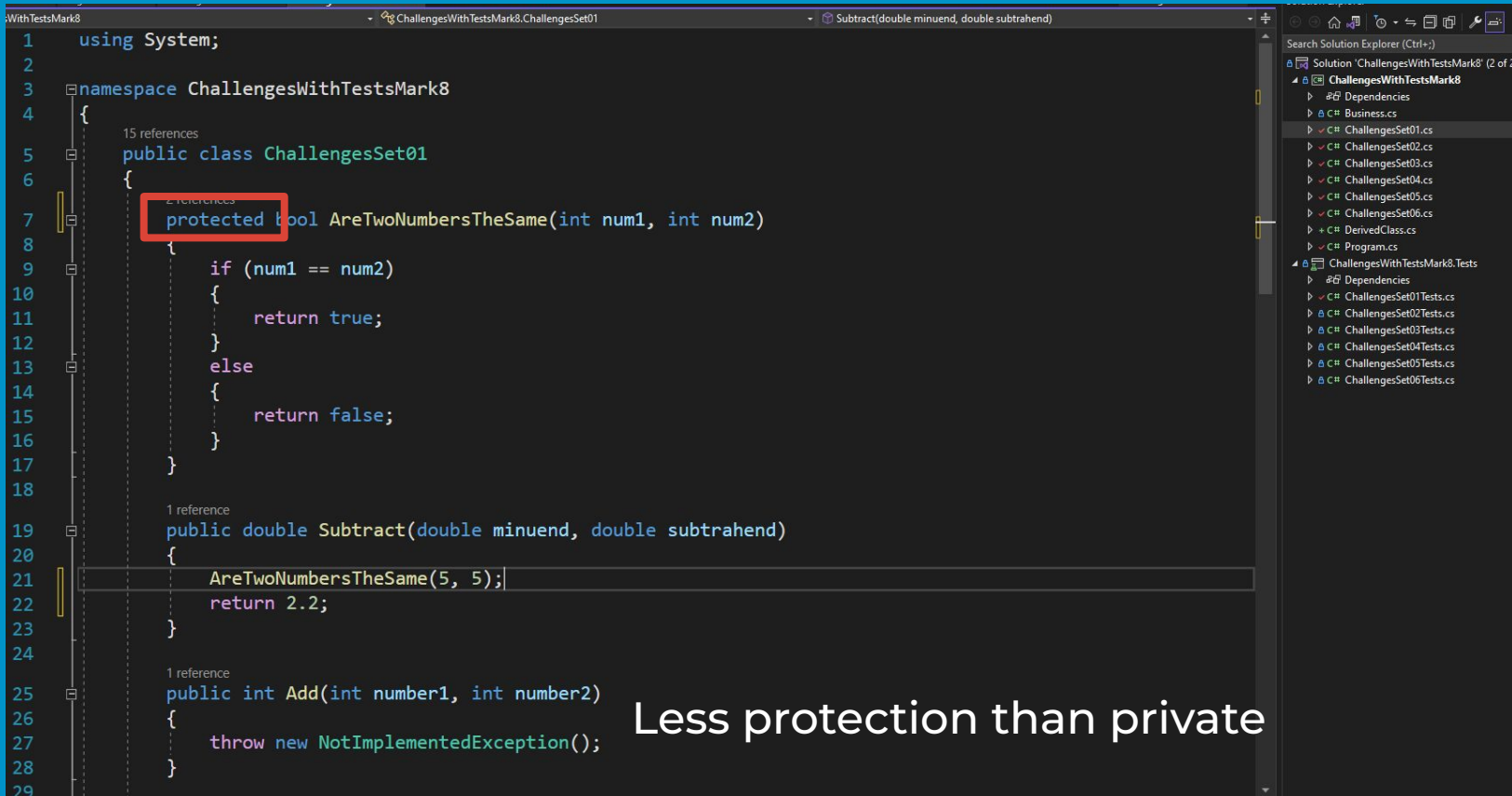
Search Solution Explorer (Ctrl+):

- Solution 'ChallengesWithTestsMark8' (2 of 2 projects)
- ChallengesWithTestsMark8
 - Dependencies
 - Business.cs
 - ChallengesSet01.cs
 - ChallengesSet02.cs
 - ChallengesSet03.cs
 - ChallengesSet04.cs
 - ChallengesSet05.cs
 - ChallengesSet06.cs
 - DerivedClass.cs
 - Program.cs
 - ChallengesWithTestsMark8.Tests
 - ChallengesSet01Tests.cs
 - ChallengesSet02Tests.cs
 - ChallengesSet03Tests.cs
 - ChallengesSet04Tests.cs
 - ChallengesSet05Tests.cs
 - ChallengesSet06Tests.cs

Protected

- protected - the member can only be accessed by code in the same class or in a derived class

Protected – Same Class or Derived Class



The screenshot shows a Visual Studio IDE with a C# code file named `ChallengesWithTestsMark8.ChallengesSet01`. The code is as follows:

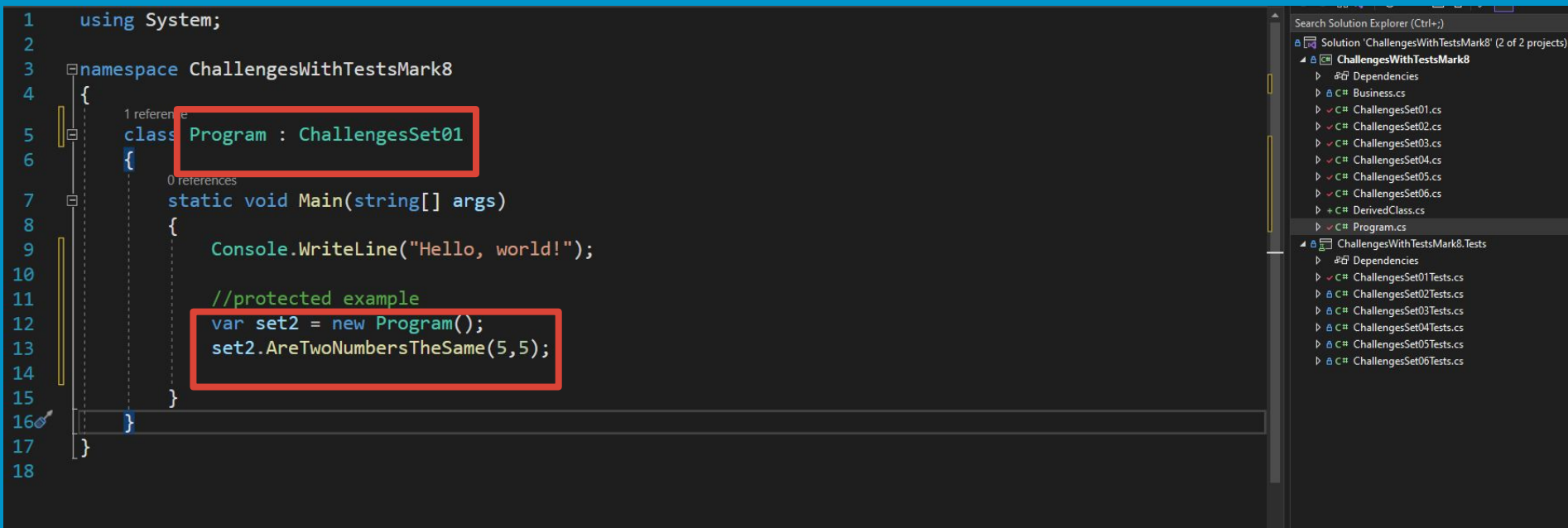
```
1 using System;
2
3 namespace ChallengesWithTestsMark8
4 {
5     15 references
6     public class ChallengesSet01
7     {
8         2 references
9         protected bool AreTwoNumbersTheSame(int num1, int num2)
10         {
11             if (num1 == num2)
12             {
13                 return true;
14             }
15             else
16             {
17                 return false;
18             }
19         }
20
21         1 reference
22         public double Subtract(double minuend, double subtrahend)
23         {
24             AreTwoNumbersTheSame(5, 5);
25             return 2.2;
26         }
27
28         1 reference
29         public int Add(int number1, int number2)
30         {
31             throw new NotImplementedException();
32         }
33     }
34 }
```

The `protected` keyword in the `AreTwoNumbersTheSame` method signature is highlighted with a red box. The `Subtract` method calls `AreTwoNumbersTheSame(5, 5);`, demonstrating that the method is accessible within the same class.

On the right side, the Solution Explorer shows the project structure for `ChallengesWithTestsMark8`, including `ChallengesSet01.cs`, `ChallengesSet02.cs`, `ChallengesSet03.cs`, `ChallengesSet04.cs`, `ChallengesSet05.cs`, `ChallengesSet06.cs`, `DerivedClass.cs`, and `Program.cs`. Below it, the `ChallengesWithTestsMark8.Tests` project is also visible, containing test files for each challenge set.

Less protection than private

Protected – Same Class or Derived Class



The screenshot displays a Visual Studio IDE with a C# code file and the Solution Explorer on the right. The code is as follows:

```
1 using System;
2
3 namespace ChallengesWithTestsMark8
4 {
5     class Program : ChallengesSet01
6     {
7         static void Main(string[] args)
8         {
9             Console.WriteLine("Hello, world!");
10
11             //protected example
12             var set2 = new Program();
13             set2.AreTwoNumbersTheSame(5,5);
14         }
15     }
16 }
17
18
```

In the code, two areas are highlighted with red boxes: the class declaration `class Program : ChallengesSet01` on line 5, and the instantiation `var set2 = new Program();` on line 12. The Solution Explorer on the right shows a project named 'ChallengesWithTestsMark8' containing several files, including `Program.cs` and `ChallengesSet01Tests.cs`.

Program is the derived class

Protected – Same Class or Derived Class

```
1 using System;
2
3 namespace ChallengesWithTestsMark8
4 {
5     1 reference
6     class Program
7     {
8         0 references
9         static void Main(string[] args)
10         {
11             Console.WriteLine("Hello, world!");
12
13             //protected example
14             var set2 = new Program();
15             set2.AreTwoNumbersTheSame(5,5);
16         }
17     }
18 }
```

Error

Solution Explorer (Ctrl+;) shows the project structure:

- Solution 'ChallengesWithTestsMark8' (2 of 2 projects)
- ChallengesWithTestsMark8
 - Dependencies
 - Business.cs
 - ChallengesSet01.cs
 - ChallengesSet02.cs
 - ChallengesSet03.cs
 - ChallengesSet04.cs
 - ChallengesSet05.cs
 - ChallengesSet06.cs
 - DerivedClass.cs
 - Program.cs
- ChallengesWithTestsMark8.Tests
 - Dependencies
 - ChallengesSet01Tests.cs
 - ChallengesSet02Tests.cs
 - ChallengesSet03Tests.cs
 - ChallengesSet04Tests.cs
 - ChallengesSet05Tests.cs
 - ChallengesSet06Tests.cs

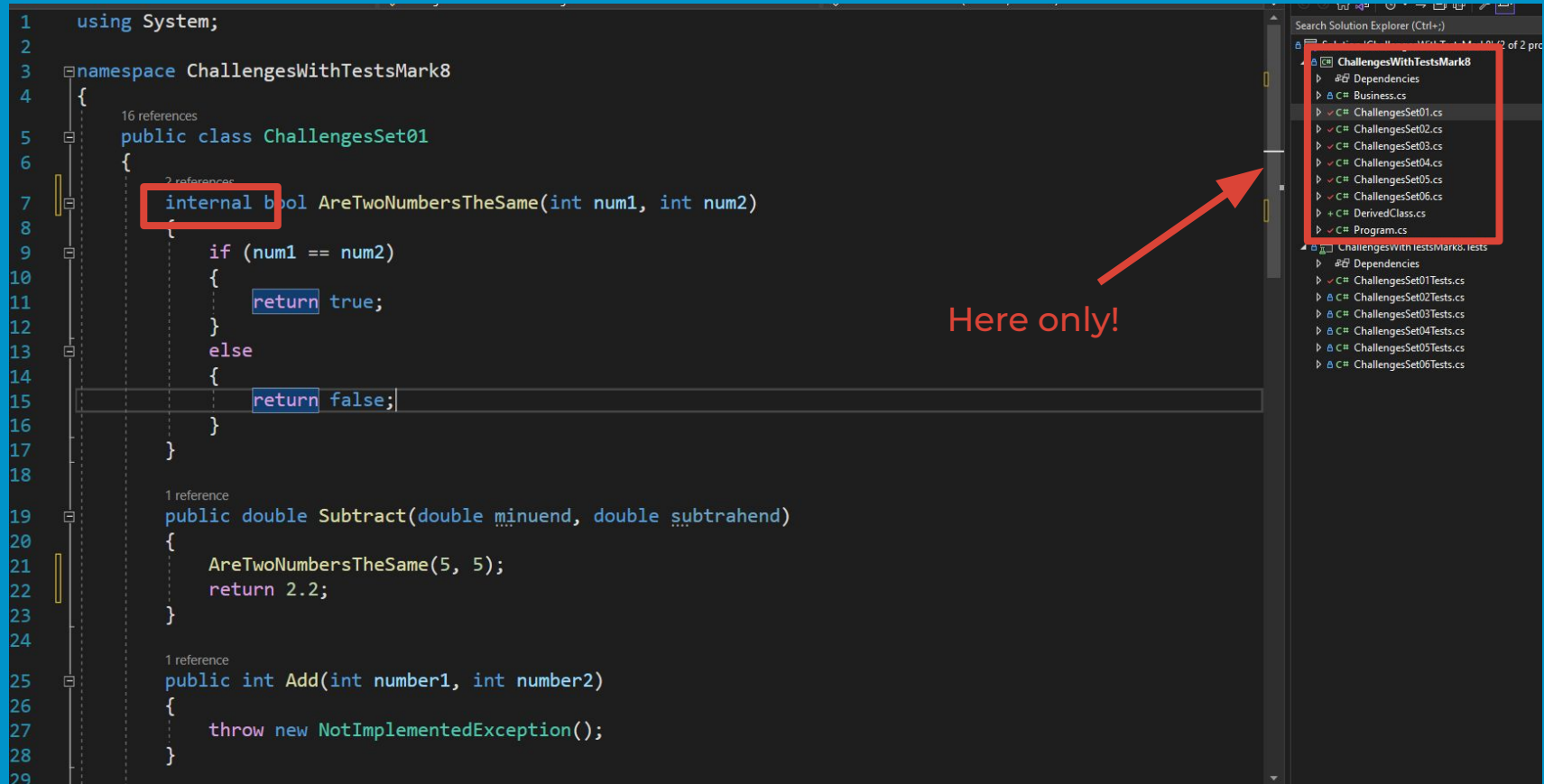
Program no longer derives from ChallengeSet01. No access!

Internal

- internal - the member can be accessed by any code in the same assembly, but not from another assembly



Internal - Only inside the project file!



The image shows a Visual Studio editor window with C# code and a Solution Explorer on the right. The code is as follows:

```
1 using System;
2
3 namespace ChallengesWithTestsMark8
4 {
5     16 references
6     public class ChallengesSet01
7     {
8         2 references
9         internal bool AreTwoNumbersTheSame(int num1, int num2)
10         {
11             if (num1 == num2)
12             {
13                 return true;
14             }
15             else
16             {
17                 return false;
18             }
19         }
20
21         1 reference
22         public double Subtract(double minuend, double subtrahend)
23         {
24             AreTwoNumbersTheSame(5, 5);
25             return 2.2;
26         }
27
28         1 reference
29         public int Add(int number1, int number2)
30         {
31             throw new NotImplementedException();
32         }
33     }
34 }
```

The word `internal` in line 9 is highlighted with a red box. In the Solution Explorer on the right, the project `ChallengesWithTestsMark8` is expanded, and its files are listed. A red box highlights the files `ChallengesSet01.cs` through `Program.cs`. A red arrow points from the text "Here only!" to the `internal` keyword in the code.

Here only!

Internal - Cannot access in diff project file!

```
1 using System;
2 using Xunit;
3
4 namespace ChallengesWithTestsMark8.Tests
5 {
6     0 references
7     public class ChallengesSet01Tests
8     {
9         [Theory]
10         [InlineData(1, 1, true)]
11         [InlineData(10, 10, true)]
12         [InlineData(99, 99, true)]
13         [InlineData(-10, -10, true)]
14         [InlineData(-1, -1, true)]
15         [InlineData(0, 1, false)]
16         [InlineData(4, 7, false)]
17         [InlineData(-1, 1, false)]
18         [InlineData(5, 6, false)]
19         0 references
20         public void AreTwoNumbersTheSameTest(int number1, int number2, bool expected)
21         {
22             // Arrange
23             ChallengesSet01 challenger = new ChallengesSet01();
24
25             // Act
26             bool actual = challenger.AreTwoNumbersTheSame(number1, number2);
27
28             // Assert
29             Assert.Equal(expected, actual);
30         }
31     }
32 }
```

Error

CS0122: 'ChallengesSet01.AreTwoNumbersTheSame(int num1, int num2)' is inaccessible due to its protection level

Show potential fixes (Alt+Enter or Ctrl+.)

Search Solution Explorer (Ctrl+.)

- Solution 'ChallengesWithTestsMark8' (2 of 2)
- ChallengesWithTestsMark8
 - # Dependencies
 - C# Business.cs
 - ✓ C# ChallengesSet01.cs
 - ✓ C# ChallengesSet02.cs
 - ✓ C# ChallengesSet03.cs
 - ✓ C# ChallengesSet04.cs
 - ✓ C# ChallengesSet05.cs
 - ✓ C# ChallengesSet06.cs
 - + C# DerivedClass.cs
 - ✓ C# Program.cs
- ChallengesWithTestsMark8.Tests
 - # Dependencies
 - C# ChallengesSet01Tests.cs
 - C# ChallengesSet02Tests.cs
 - C# ChallengesSet03Tests.cs
 - C# ChallengesSet04Tests.cs
 - C# ChallengesSet05Tests.cs
 - C# ChallengesSet06Tests.cs

Protected Internal

- protected internal - the member can be accessed by any code in the same assembly, or by any derived class in another assembly

AKA same project file OR derived class in a different project file

```
// Assembly A
using System;

namespace MyNamespace
{
    protected internal class ProtectedInternalClass
    {
        public void SomeMethod()
        {
            Console.WriteLine("Protected Internal Class");
        }
    }
}

// Assembly B
using MyNamespace;

namespace AnotherNamespace
{
    public class DerivedClass : ProtectedInternalClass
    {
        public void AnotherMethod()
        {
            Console.WriteLine("Derived Class");
        }
    }

    public class Program
    {
        static void Main(string[] args)
        {
            DerivedClass derivedObj = new DerivedClass();
            derivedObj.SomeMethod();    // Accessing protected internal method
            derivedObj.AnotherMethod(); // Accessing public method
        }
    }
}
```

🔄 Regenerate response

Access Levels

Summary table

Caller's location	public	protected internal	protected	internal	private protected	private
Within the class	✓	✓	✓	✓	✓	✓
Derived class (same assembly)	✓	✓	✓	✓	✓	✗
Non-derived class (same assembly)	✓	✓	✗	✓	✗	✗
Derived class (different assembly)	✓	✓	✓	✗	✗	✗
Non-derived class (different assembly)	✓	✗	✗	✗	✗	✗

Bank Encapsulation Exercise Bonus

- Create withdraw method
- Create welcome screen where user can pick deposit, withdraw, or balance
- Set pin number
- Get user input for deposit and withdraw methods
- Exit out of the program when user chooses



Bank Encapsulation Exercise