C# has six combinations of access modifier keywords:

- public: accessible from anywhere in the program.
- private: accessible only within the class where it is defined.
- protected: accessible within the class where it is defined, and in derived classes.
- internal: accessible only within the same assembly.
- protected internal: accessible within the same assembly, and in derived classes.
- private protected: accessible within the same class and in derived classes, but only within the same assembly.

The static keyword means that a member belongs to the type itself, rather than to instances of the type. The const keyword is used to define a compile-time constant value that cannot be changed. The readonly keyword is used to declare a value that can be set only once, typically in the constructor or initialization block of the class.

A constructor is a special method in a class that is used to initialize objects of that class. It typically has the same name as the class, and may take one or more parameters to initialize the object's fields.

The partial keyword is useful because it allows a class, struct, or interface to be defined in multiple files. This can make it easier to organize large or complex classes, and can allow different developers to work on different parts of a class without interfering with each other.

A tuple is a lightweight data structure in C# that allows multiple values of different types to be stored together in a single object. Tuples can be useful when a method needs to return multiple values, or when multiple values need to be passed as parameters to a method.

The C# record keyword is used to define classes that are intended primarily to store data, rather than to encapsulate behavior. Records can be used to define immutable data structures with automatically generated methods for comparison, hashing, and string representation.

Overloading means defining multiple methods or constructors with the same name but different parameter lists. Overloading allows the same method name to be used for different purposes. Overriding means providing a new implementation for a method that is already defined in a base class or interface. Overriding allows a subclass to provide a different implementation of a method that is inherited from its parent class or interface.

A field is a variable that belongs to an instance of a class or struct, and can be accessed directly by the class or struct. A property is a member of a class or struct that provides a way to access or modify a private field, while also allowing the class to enforce constraints or perform additional logic on the value.

To make a method parameter optional, you can give it a default value. This allows the method to be called with or without that parameter, and if the parameter is omitted, the default value is used.

An interface is a type that defines a set of methods and properties that a class must implement. An interface provides a way to define a contract between different classes, allowing them to communicate with each other in a standardized way. An abstract class is a class that cannot be instantiated, and is used as a base class for other classes. An abstract class can provide a partial implementation of a class, including abstract methods that must be implemented by subclasses. The main difference between an interface and an abstract class is that an interface defines only the methods and properties that must be implemented, while an abstract class can also provide some implementation.

Members of an interface are by default public and abstract.

True

True

True

True

True

True

True

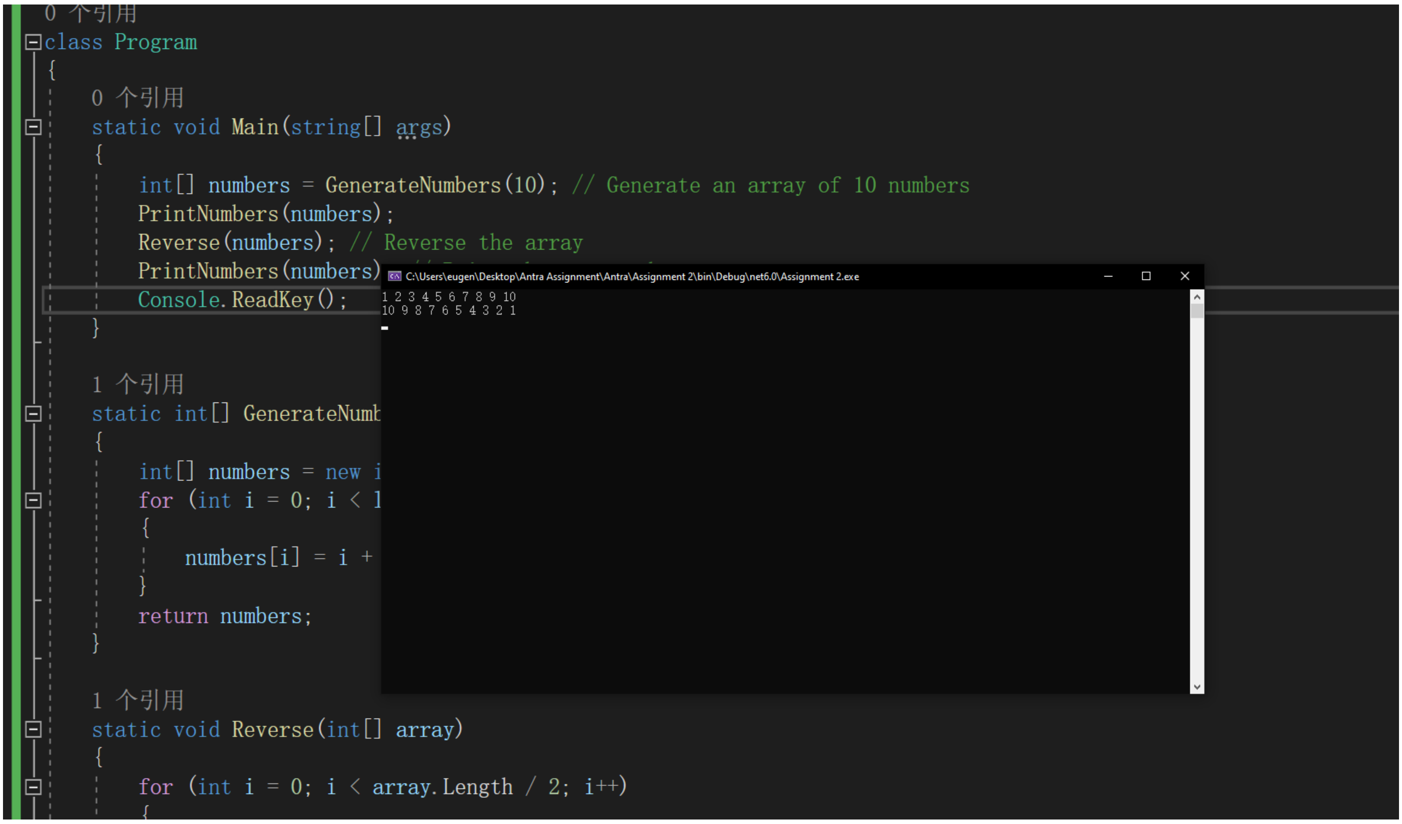
False

False

False

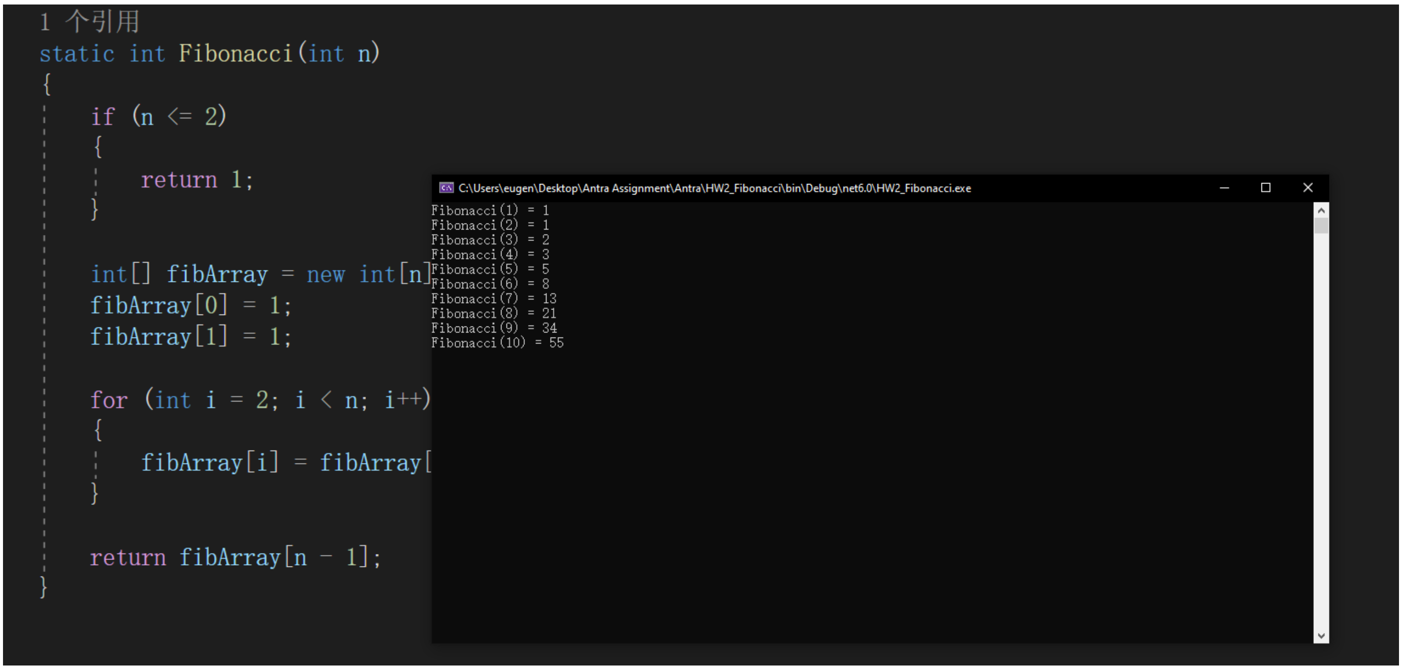
False

True


C:\Users\eugen\Desktop\Antra Assignment\Antra\Assignment 2\bin\Debug\net6.0\Assignment 2.exe

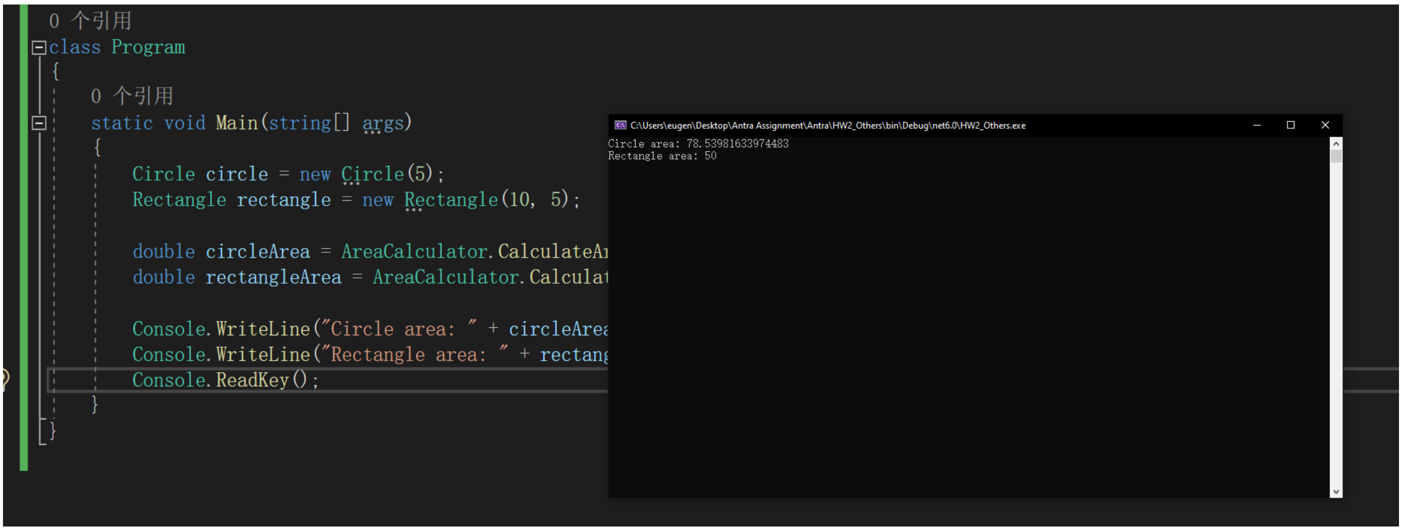
```
0 个引用
class Program
{
    0 个引用
    static void Main(string[] args)
    {
        int[] numbers = GenerateNumbers(10); // Generate an array of 10 numbers
        PrintNumbers(numbers);
        Reverse(numbers); // Reverse the array
        PrintNumbers(numbers)
        Console.ReadKey();
    }

    1 个引用
    static int[] GenerateNumb
    {
        int[] numbers = new i
        for (int i = 0; i < 1
        {
            numbers[i] = i +
        }
        return numbers;
    }

    1 个引用
    static void Reverse(int[] array)
    {
        for (int i = 0; i < array.Length / 2; i++)
        {
```

```
1 2 3 4 5 6 7 8 9 10
10 9 8 7 6 5 4 3 2 1
```


C:\Users\eugen\Desktop\Antra Assignment\Antra\HW2_Fibonacci\bin\Debug\net6.0\HW2_Fibonacci.exe

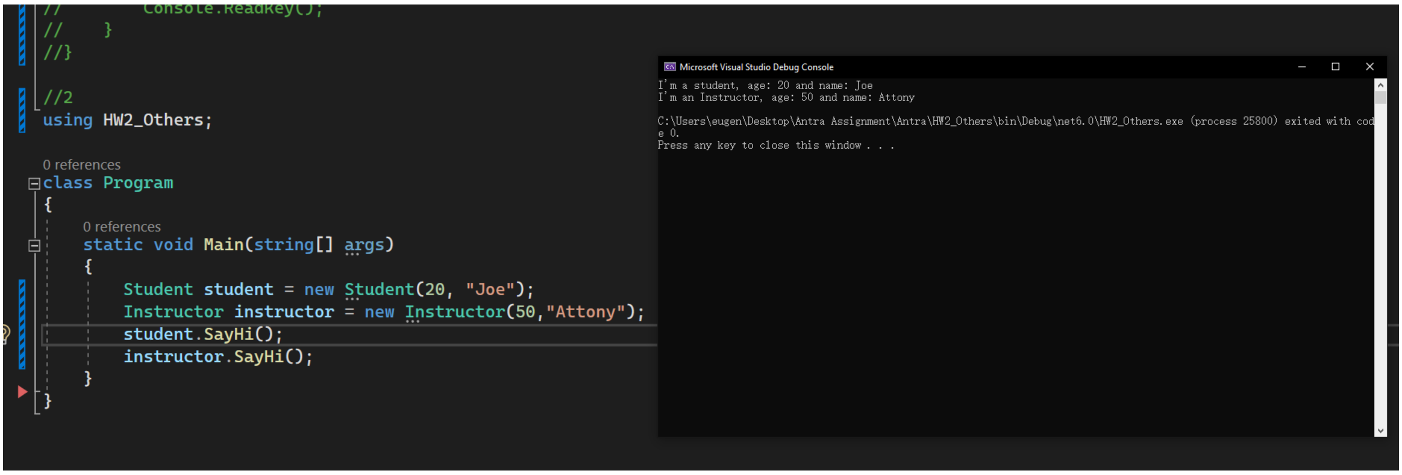
```
1 个引用
static int Fibonacci(int n)
{
    if (n <= 2)
    {
        return 1;
    }

    int[] fibArray = new int[n]
    fibArray[0] = 1;
    fibArray[1] = 1;

    for (int i = 2; i < n; i++)
    {
        fibArray[i] = fibArray[
    }

    return fibArray[n - 1];
}
```

```
Fibonacci(1) = 1
Fibonacci(2) = 1
Fibonacci(3) = 2
Fibonacci(4) = 3
Fibonacci(5) = 5
Fibonacci(6) = 8
Fibonacci(7) = 13
Fibonacci(8) = 21
Fibonacci(9) = 34
Fibonacci(10) = 55
```


C:\Users\eugen\Desktop\Antra Assignment\Antra\HW2_Others\bin\Debug\net6.0\HW2_Others.exe

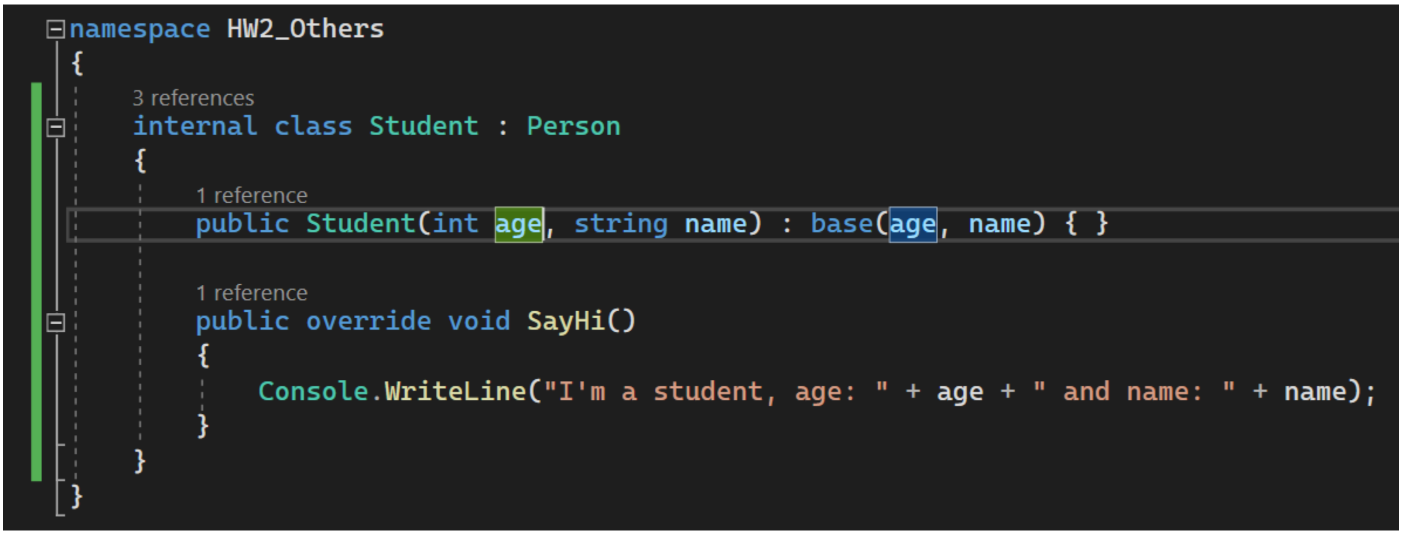
```
0 个引用
class Program
{
    0 个引用
    static void Main(string[] args)
    {
        Circle circle = new Circle(5);
        Rectangle rectangle = new Rectangle(10, 5);

        double circleArea = AreaCalculator.CalculateAr
        double rectangleArea = AreaCalculator.Calculat

        Console.WriteLine("Circle area: " + circleArea
        Console.WriteLine("Rectangle area: " + rectang
        Console.ReadKey();
    }
}
```

```
Circle area: 78.53981633974483
Rectangle area: 50
```

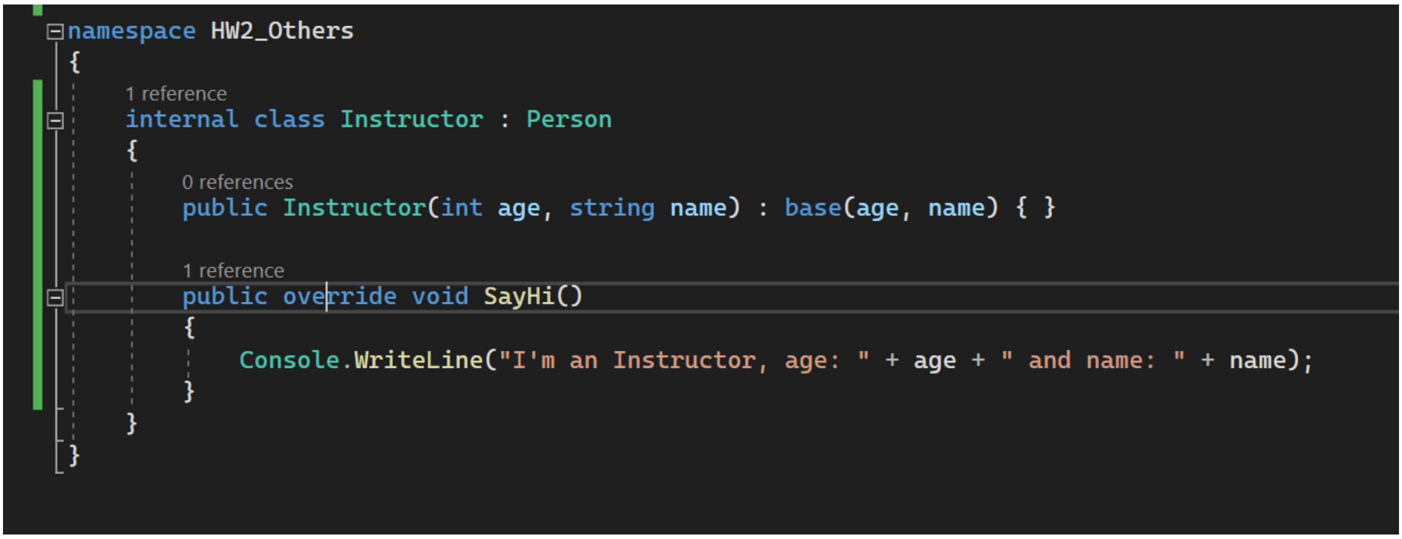


```
//            Console.ReadKey();
//    }
//}

//2
using HW2_Others;

0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Student student = new Student(20, "Joe");
        Instructor instructor = new Instructor(50,"Attony");
        student.SayHi();
        instructor.SayHi();
    }
}
```

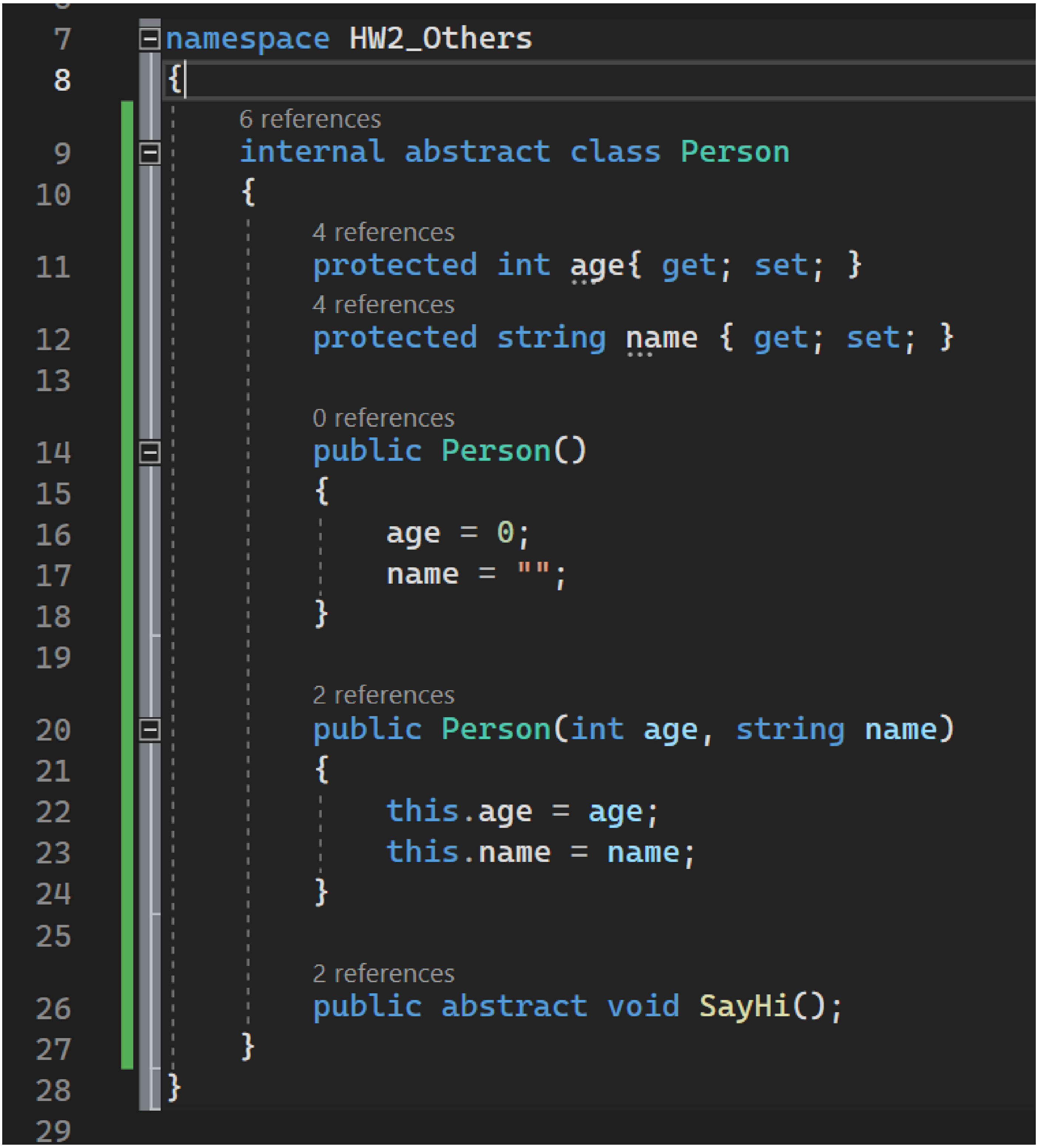
```
I'm a student, age: 20 and name: Joe
I'm an Instructor, age: 50 and name: Attony

C:\Users\eugen\Desktop\Antra Assignment\Antra\HW2_Others\bin\Debug\net6.0\HW2_Others.exe (process 25800) exited with cod
e 0.
Press any key to close this window . . .
```



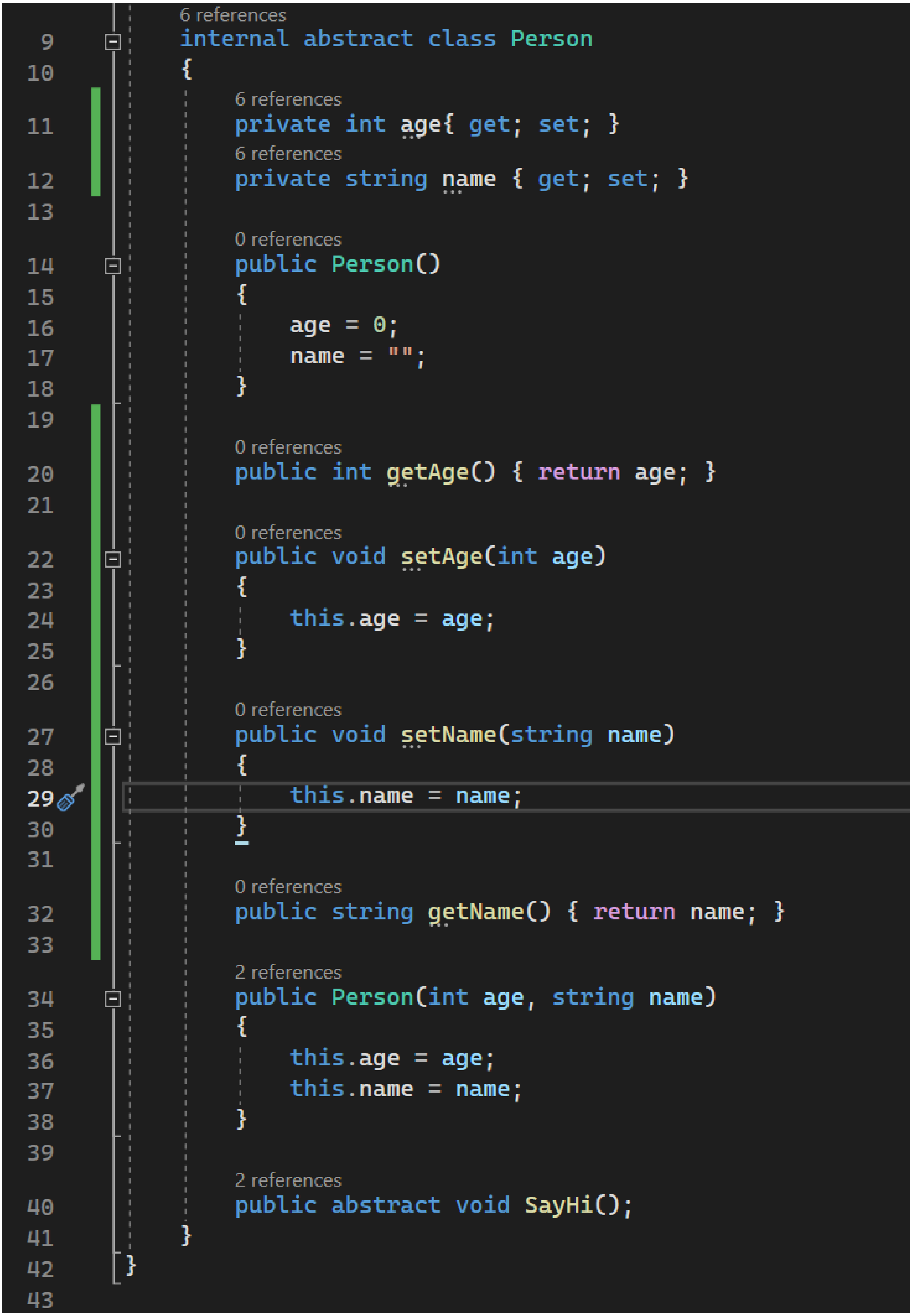
```
namespace HW2_Others
{
    3 references
    internal class Student : Person
    {
        1 reference
        public Student(int age, string name) : base(age, name) { }

        1 reference
        public override void SayHi()
        {
            Console.WriteLine("I'm a student, age: " + age + " and name: " + name);
        }
    }
}
```



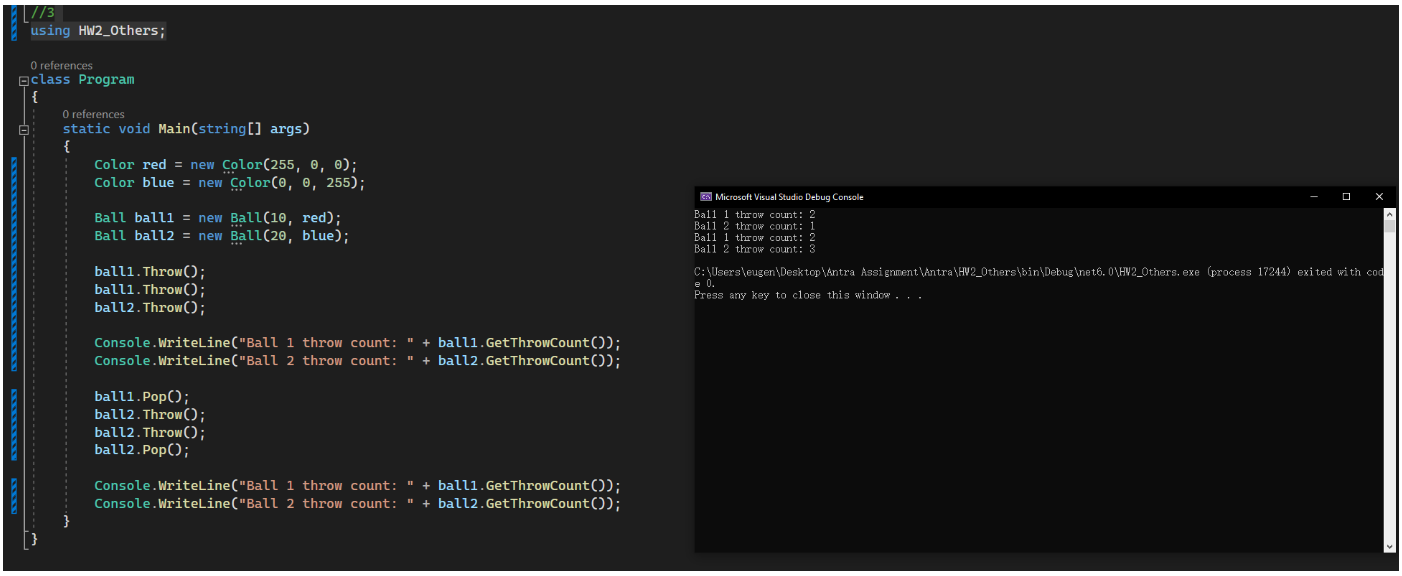
```
namespace HW2_Others
{
    1 reference
    internal class Instructor : Person
    {
        0 references
        public Instructor(int age, string name) : base(age, name) { }

        1 reference
        public override void SayHi()
        {
            Console.WriteLine("I'm an Instructor, age: " + age + " and name: " + name);
        }
    }
}
```



```
namespace HW2_Others
{
        6 references
        internal abstract class Person
        {
            4 references
            protected int age{ get; set; }
            4 references
            protected string name { get; set; }

            0 references
            public Person()
            {
                age = 0;
                name = "";
            }

            2 references
            public Person(int age, string name)
            {
                this.age = age;
                this.name = name;
            }

            2 references
            public abstract void SayHi();
        }
}
```

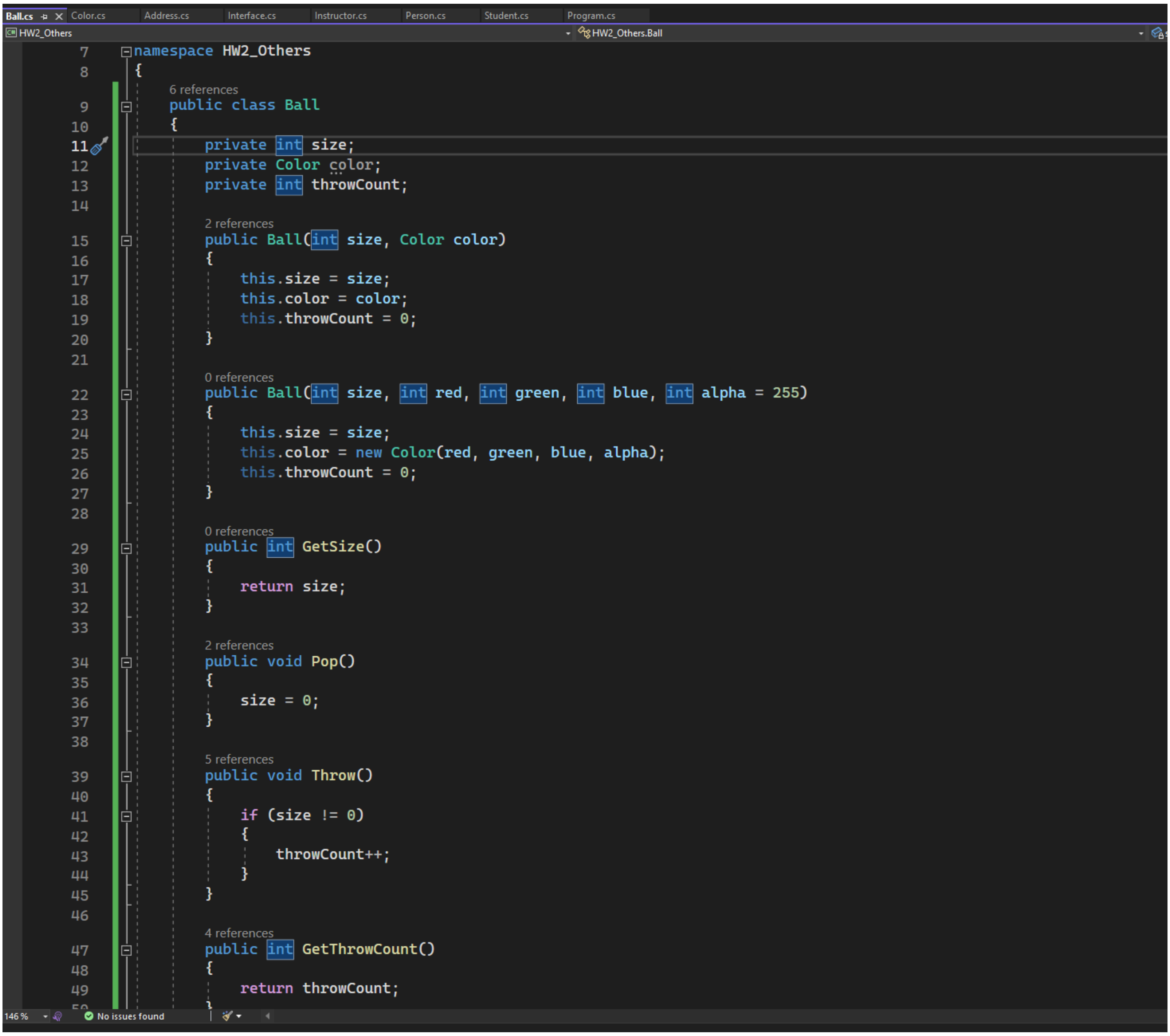


```
    6 references
    internal abstract class Person
    {
        6 references
        private int age{ get; set; }
        6 references
        private string name { get; set; }

        0 references
        public Person()
        {
            age = 0;
            name = "";
        }

        0 references
        public int getAge() { return age; }

        0 references
        public void setAge(int age)
        {
            this.age = age;
        }

        0 references
        public void setName(string name)
        {
            this.name = name;
        }

        0 references
        public string getName() { return name; }

        2 references
        public Person(int age, string name)
        {
            this.age = age;
            this.name = name;
        }

        2 references
        public abstract void SayHi();
    }
}
```

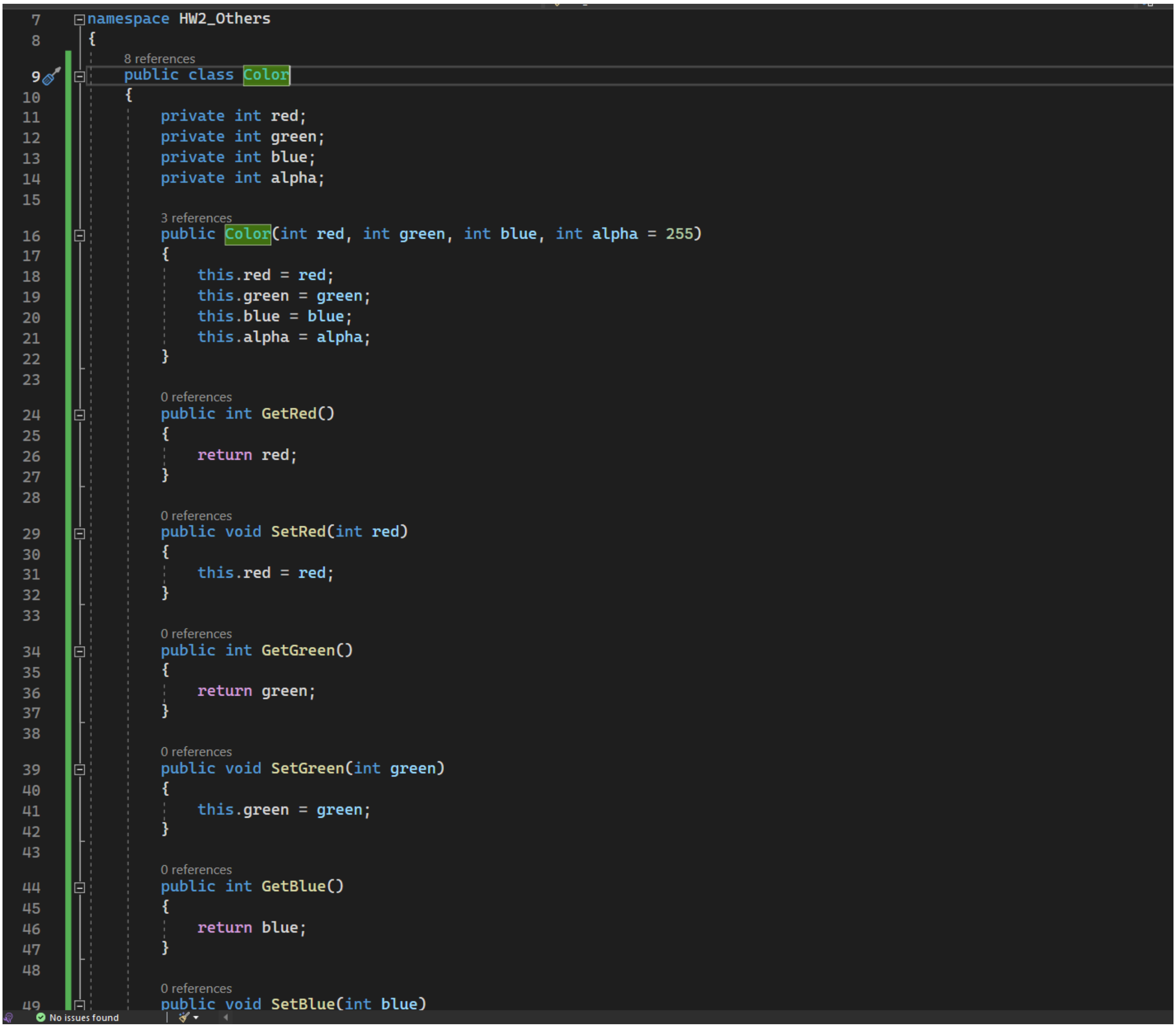


```csharp
//3
using HW2_Others;

class Program
{
    static void Main(string[] args)
    {
        Color red = new Color(255, 0, 0);
        Color blue = new Color(0, 0, 255);

        Ball ball1 = new Ball(10, red);
        Ball ball2 = new Ball(20, blue);

        ball1.Throw();
        ball1.Throw();
        ball2.Throw();

        Console.WriteLine("Ball 1 throw count: " + ball1.GetThrowCount());
        Console.WriteLine("Ball 2 throw count: " + ball2.GetThrowCount());

        ball1.Pop();
        ball2.Throw();
        ball2.Throw();
        ball2.Pop();

        Console.WriteLine("Ball 1 throw count: " + ball1.GetThrowCount());
        Console.WriteLine("Ball 2 throw count: " + ball2.GetThrowCount());
    }
}
```

```
Ball 1 throw count: 2
Ball 2 throw count: 1
Ball 1 throw count: 2
Ball 2 throw count: 3

C:\Users\eugen\Desktop\Antra Assignment\Antra\HW2_Others\bin\Debug\net6.0\HW2_Others.exe (process 17244) exited with cod
e 0.
Press any key to close this window . . .
```



```csharp
namespace HW2_Others
{
    public class Ball
    {
        private int size;
        private Color color;
        private int throwCount;

        public Ball(int size, Color color)
        {
            this.size = size;
            this.color = color;
            this.throwCount = 0;
        }

        public Ball(int size, int red, int green, int blue, int alpha = 255)
        {
            this.size = size;
            this.color = new Color(red, green, blue, alpha);
            this.throwCount = 0;
        }

        public int GetSize()
        {
            return size;
        }

        public void Pop()
        {
            size = 0;
        }

        public void Throw()
        {
            if (size != 0)
            {
                throwCount++;
            }
        }

        public int GetThrowCount()
        {
            return throwCount;
```



```
namespace HW2_Others
{
    8 references
    public class Color
    {
        private int red;
        private int green;
        private int blue;
        private int alpha;

        3 references
        public Color(int red, int green, int blue, int alpha = 255)
        {
            this.red = red;
            this.green = green;
            this.blue = blue;
            this.alpha = alpha;
        }

        0 references
        public int GetRed()
        {
            return red;
        }

        0 references
        public void SetRed(int red)
        {
            this.red = red;
        }

        0 references
        public int GetGreen()
        {
            return green;
        }

        0 references
        public void SetGreen(int green)
        {
            this.green = green;
        }

        0 references
        public int GetBlue()
        {
            return blue;
        }

        0 references
        public void SetBlue(int blue)
```