گزارش پروژه دوم

نام و نام خانوادگی

نگار فتحی

شماره دانشجویی

۹۷۷۲۳۱۳۷

استاد

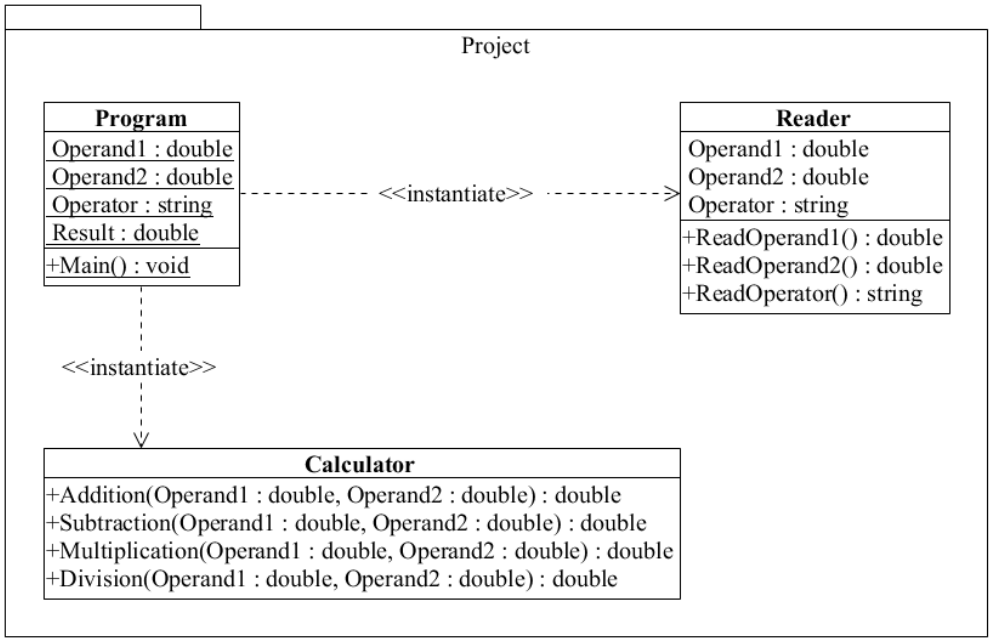
دکتر سعید پارسا

درس

کامپایلر پیشرفته

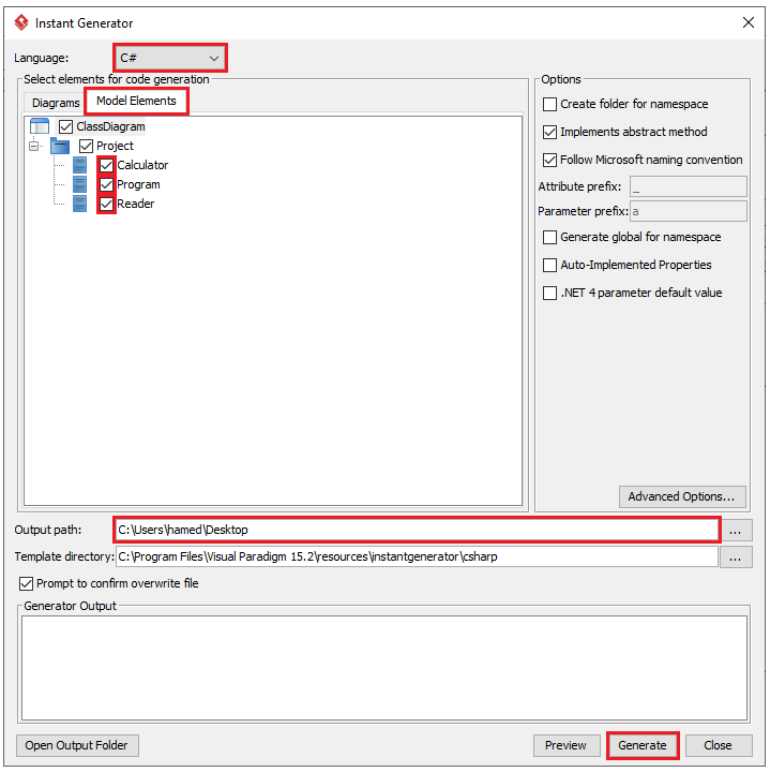
در پروژه انجام‌شده، بعداز رسم نمودار کلاس در محیط Visual Paradigm، تولید کد C# از روی نمودار کلاس، ایجاد پروژه در محیط Visual Studio، افزودن کلاس‌ها به پروژه و درنهایت پیاده‌سازی کلاس‌ها، متدهای برنامه به‌کمک ابزار NUnit تحت آزمون واحد قرار گرفتند.

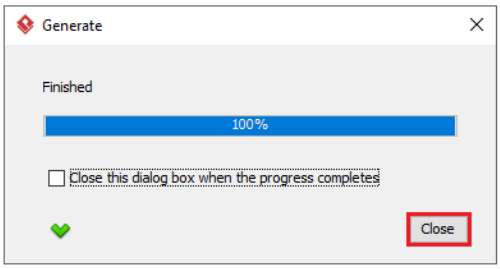
رسم نمودار کلاس در محیط Visual Paradigm:



تولید کد C# از روی نمودار کلاس:

Visual Paradigm ⇒ Code ⇒ Instant Generator ⇒





کد تولیدشده برای کلاس Program:

```
using System;
namespace Project {
	public class Program {
		private static double operand1;
		private static double operand2;
		private static string operator;
		private static double result;

		public static void Main() {
			throw new System.Exception("Not implemented");
		}

	}

}
```

کد تولیدشده برای کلاس Reader:

```
using System;
namespace Project {
	public class Reader {
		private double operand1;
		private double operand2;
		private string operator;

		public double ReadOperand1() {
			throw new System.Exception("Not implemented");
		}
		public double ReadOperand2() {
			throw new System.Exception("Not implemented");
		}
		public string ReadOperator() {
			throw new System.Exception("Not implemented");
		}

	}

}
```

کد تولیدشده برای کلاس Calculator:

```
using System;
namespace Project {
	public class Calculator {
		public double Addition(double operand1, double operand2) {
			throw new System.Exception("Not implemented");
```

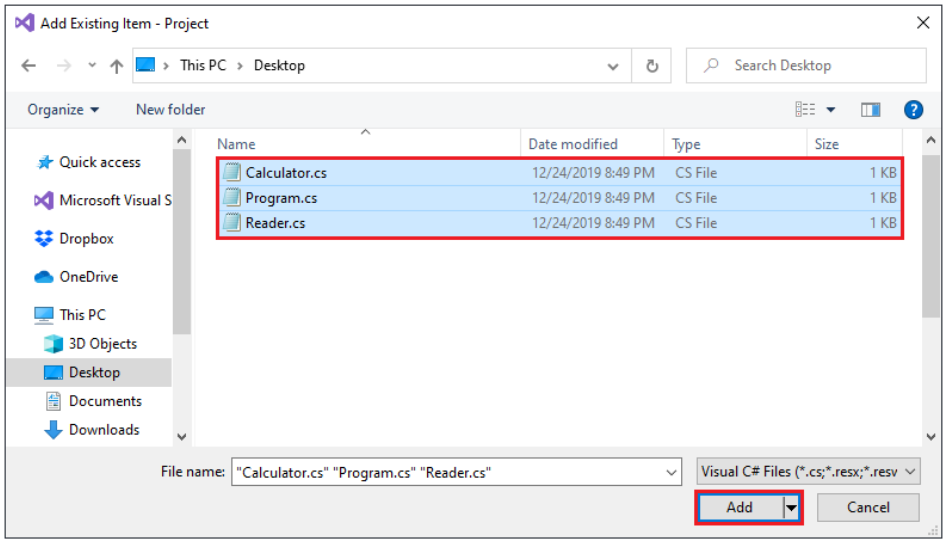
```
                }
                public double Subtraction(double operand1, double operand2) {
                        throw new System.Exception("Not implemented");
                }
                public double Multiplication(double operand1, double operand2) {
                        throw new System.Exception("Not implemented");
                }
                public double Division(double operand1, double operand2) {
                        throw new System.Exception("Not implemented");
                }

        }

}
```

ایجاد پروژه در محیط Visual Studio:

Visual Studio ⇒ File ⇒ New ⇒ Project ⇒ Visual C# ⇒ Empty Project (.NET Framework) ⇒ Neme: Project ⇒ Framework: .NET Framework 4.6.1 ⇒ Ok.

افزودن کلاس‌ها به پروژه:

Visual Studio ⇒ Solution Explorer ⇒ Project ⇒ Add ⇒ Existing Item... ⇒



پیاده‌سازی کلاس‌ها:

پیاده‌سازی کلاس Program:

```
using System;

namespace Project
{
   public class Program
   {
      private static double operand1;
      private static double operand2;
      private static string Operator;
```

```
        private static double result;
        public static void Main()
        {
            Reader reader = new Reader();
            operand1 = reader.ReadOperand1();
            operand2 = reader.ReadOperand2();
            Operator = reader.ReadOperator();
            Calculator calculator = new Calculator();
            switch (Operator)
            {
                case "+":
                    result = calculator.Addition(operand1, operand2);
                    break;
                case "-":
                    result = calculator.Subtraction(operand1, operand2);
                    break;
                case "*":
                    result = calculator.Multiplication(operand1, operand2);
                    break;
                case "/":
                    result = calculator.Division(operand1, operand2);
                    break;
            }
            Console.Write("Result: " + result);
            Console.ReadLine();
        }
    }
}
```

پیاده‌سازی کلاس Reader:

```
using System;

namespace Project
{
    public class Reader
    {
        private double operand1;
        private double operand2;
        private string Operator;
        public double ReadOperand1()
        {
            Console.Write("Operand1: ");
            operand1 = double.Parse(Console.ReadLine());
            return operand1;
        }
        public double ReadOperand2()
        {
            Console.Write("Operand2: ");
            operand2 = double.Parse(Console.ReadLine());
            return operand2;
        }
        public string ReadOperator()
```

```
    {
        Console.Write("Operator: ");
        Operator = Console.ReadLine();
        return Operator;
    }
  }
}
```

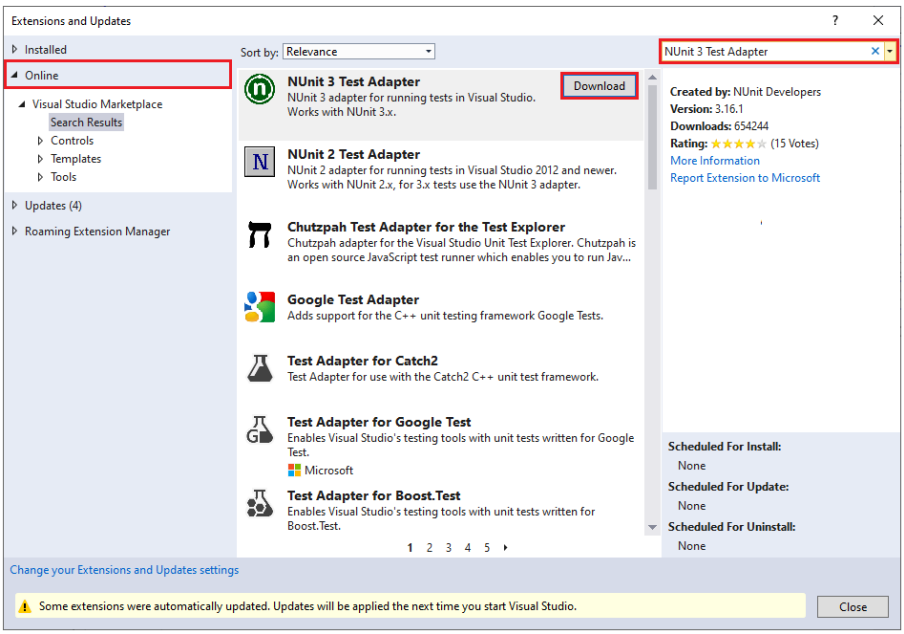
پیاده‌سازی کلاس Calculator:

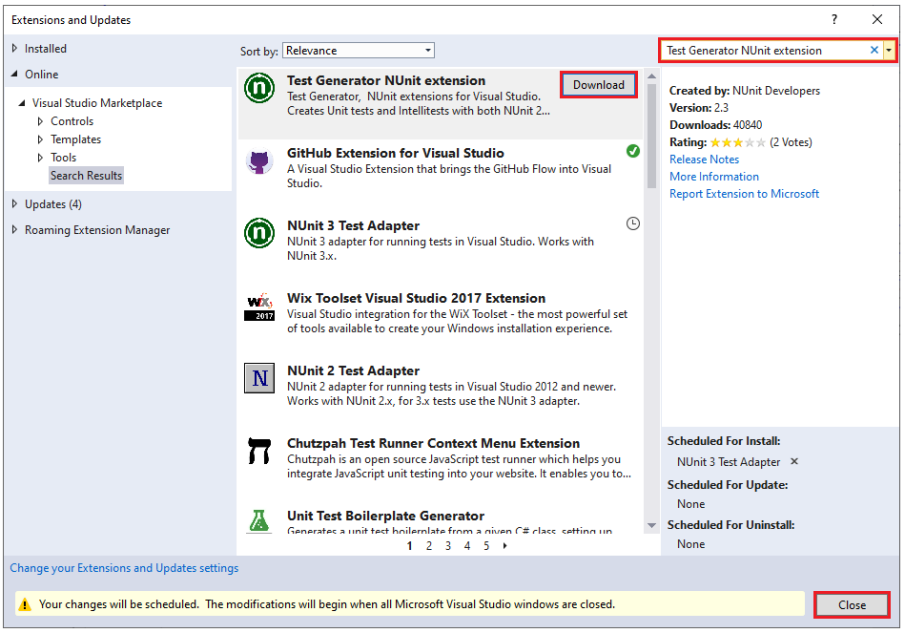
```
using System;

namespace Project
{
  public class Calculator
  {
    public double Addition(double operand1, double operand2)
    {
        return operand1 + operand2;
    }
    public double Subtraction(double operand1, double operand2)
    {
        return operand1 - operand2;
    }
    public double Multiplication(double operand1, double operand2)
    {
        return operand1 * operand2;
    }
    public double Division(double operand1, double operand2)
    {
        return operand1 / operand2;
    }
  }
}
```

نصب NUnit:

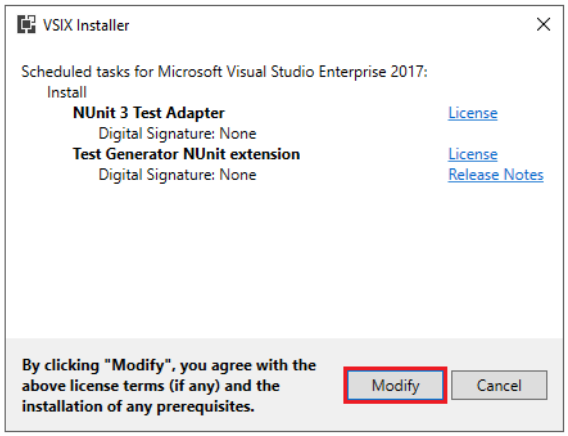
ازآن‌جایی‌که قرار است متدهای برنامه موردنظر را به‌کمک ابزار NUnit تحت آزمون واحد قرار دهیم، باید مراحل نصب آن به‌صورت زیر طی شود.

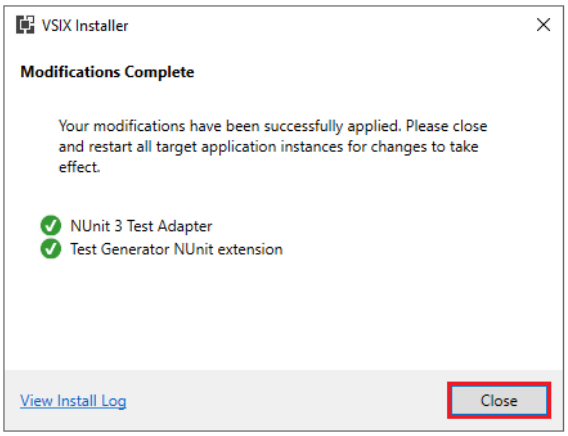
Visual Studio ⇒ Tools ⇒ Extension and Updates… ⇒





⇒ Close Visual Studio ⇒

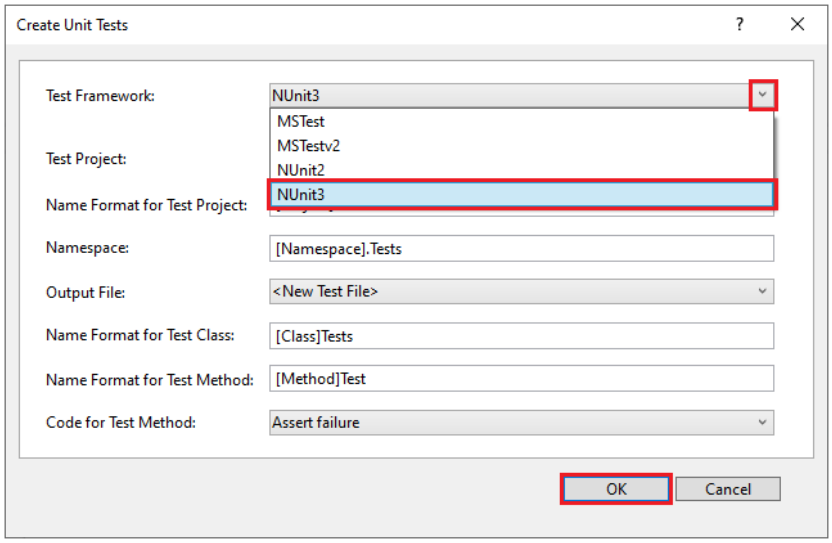




⇒ Open Visual Studio.

آزمون واحد:

در مرحله اول، قصد داریم متدهای Addition، Subtraction، Multiplication و Division کلاس Calculator را تحت آزمون واحد قرار دهیم. بدین‌منظور به‌ترتیب بر روی این متدها کلیک راست کرده و بعد از زدن گزینه Create Unit Tests مطابق شکل زیر عمل می‌کنیم.



حال در قسمت Solution Explorer پوشه‌ای به نام ProjectTests ایجاد شده‌است که حاوی یک کلاس با نام CalculatorTests.cs است. محتویات این کلاس به‌صورت زیر است.

```
using NUnit.Framework;

namespace Project.Tests
{
   [TestFixture()]
   public class CalculatorTests
   {
      [Test()]
      public void AdditionTest()
```

```
    {
      Assert.Fail();
    }
    [Test()]
    public void SubtractionTest()
    {
      Assert.Fail();
    }
    [Test()]
    public void MultiplicationTest()
    {
      Assert.Fail();
    }
    [Test()]
    public void DivisionTest()
    {
      Assert.Fail();
    }
  }
}
```

به‌منظور این‌که متدهای Addition، Subtraction، Multiplication و Division کلاس Calculator را تحت آزمون واحد قرار دهیم، متدهای AdditionTest، SubtractionTest، MultiplicationTest و DivisionTest کلاس CalculatorTests.cs را به‌صورت زیر بازنویسی می‌کنیم.

```
using NUnit.Framework;

namespace Project.Tests
{
  [TestFixture()]
  public class CalculatorTests
  {
    [Test()]
    public void AdditionTest()
    {
      Calculator calculator = new Calculator();
      double result = calculator.Addition(1, 2);
      Assert.AreEqual(3, result);
    }
    [Test()]
    public void SubtractionTest()
    {
      Calculator calculator = new Calculator();
      double result = calculator.Subtraction(1, 2);
      Assert.AreEqual(-1, result);
    }
    [Test()]
    public void MultiplicationTest()
    {
      Calculator calculator = new Calculator();
      double result = calculator.Multiplication(1, 2);
      Assert.AreEqual(2, result);
    }
    [Test()]
    public void DivisionTest()
```

```
    {
        Calculator calculator = new Calculator();
        double result = calculator.Division(1, 2);
        Assert.AreEqual(0.5, result);
    }
  }
}
```

در مرحله دوم، قصد داریم متدهای ReadOperand1، ReadOperand2 و ReadOperator کلاس Reader را تحت آزمون واحد قرار دهیم. بدین‌منظور مطابق آنچه برای متدهای کلاس Calculator گفته شد، عمل می‌کنیم. سپس در پوشه ProjectTests در قسمت Solution Explorer شاهد ایجاد کلاس دیگری با نام ReaderTests.cs خواهیم بود که محتویات آن به‌صورت زیر است.

```
using NUnit.Framework;

namespace Project.Tests
{
  [TestFixture()]
  public class ReaderTests
  {
    [Test()]
    public void ReadOperand1Test()
    {
      Assert.Fail();
    }
    [Test()]
    public void ReadOperand2Test()
    {
      Assert.Fail();
    }
    [Test()]
    public void ReadOperatorTest()
    {
      Assert.Fail();
    }
  }
}
```

این کلاس را به‌صورت زیر بازنویسی می‌کنیم.

```
using System;
using System.IO;
using NUnit.Framework;

namespace Project.Tests
{
  [TestFixture()]
  public class ReaderTests
  {
    [Test()]
    public void ReadOperand1Test()
    {
      var input = new StringReader("1");
      Console.SetIn(input);
      Reader reader = new Reader();
```

```
            double result = reader.ReadOperand1();
            Assert.AreEqual(1, result);
        }
        [Test()]
        public void ReadOperand2Test()
        {
            var input = new StringReader("2");
            Console.SetIn(input);
            Reader reader = new Reader();
            double result = reader.ReadOperand2();
            Assert.AreEqual(2, result);
        }
        [Test()]
        public void ReadOperatorTest()
        {
            var input = new StringReader("+");
            Console.SetIn(input);
            Reader reader = new Reader();
            string result = reader.ReadOperator();
            Assert.AreEqual("+", result);
        }
    }
}
```

حال قصد داریم متد Main کلاس Program را تحت آزمون واحد قرار دهیم. اما باتوجه‌به نحوه پیاده‌سازی صورت‌گرفته، این متد قابلیت آزمون ندارد. بدین‌منظور پیاده‌سازی را به صورت زیر تغییر می‌دهیم.

پیاده‌سازی جدید کلاس Program:

```
using System;

namespace Project
{
    public class MyClass
    {
        private IReader iReader;
        private ICalculator iCalculator;
        private static double operand1;
        private static double operand2;
        private static string Operator;
        private static double result;
        public MyClass(IReader iReader, ICalculator iCalculator)
        {
            this.iReader = iReader;
            this.iCalculator = iCalculator;
        }
        public void MyMethod()
        {
            operand1 = iReader.ReadOperand1();
            operand2 = iReader.ReadOperand2();
            Operator = iReader.ReadOperator();
            switch (Operator)
            {
```

```
            case "+":
               result = iCalculator.Addition(operand1, operand2);
               break;
            case "-":
               result = iCalculator.Subtraction(operand1, operand2);
               break;
            case "*":
               result = iCalculator.Multiplication(operand1, operand2);
               break;
            case "/":
               result = iCalculator.Division(operand1, operand2);
               break;
         }
         Console.Write("Result: " + result);
         Console.ReadLine();
      }
   }
   public class Program
   {
      public static void Main()
      {
         Reader reader = new Reader();
         Calculator calculator = new Calculator();
         MyClass myClass = new MyClass(reader, calculator);
         myClass.MyMethod();
      }
   }
}
```

پیاده‌سازی جدید کلاس Reader:

```
using System;

namespace Project
{
   public interface IReader
   {
      double ReadOperand1();
      double ReadOperand2();
      string ReadOperator();
   }
   public class ReaderStub : IReader
   {
      public double ReadOperand1()
      {
         return 1;
      }
      public double ReadOperand2()
      {
         return 2;
      }
      public string ReadOperator()
      {
         return "+";
```

```
    }
  }
  public class Reader : IReader
  {
    private double operand1;
    private double operand2;
    private string Operator;
    public double ReadOperand1()
    {
      Console.Write("Operand1: ");
      operand1 = double.Parse(Console.ReadLine());
      return operand1;
    }
    public double ReadOperand2()
    {
      Console.Write("Operand2: ");
      operand2 = double.Parse(Console.ReadLine());
      return operand2;
    }
    public string ReadOperator()
    {
      Console.Write("Operator: ");
      Operator = Console.ReadLine();
      return Operator;
    }
  }
}
```

در پیاده‌سازی جدید کلاس Reader، کلاسی به نام ReaderStub اضافه شده‌است که برای انجام آزمون واحد متدی که درون خود متدهای کلاس Reader را فراخوانی می‌کند، ضروری است.

پیاده‌سازی جدید کلاس Calculator:

```
using System;

namespace Project
{
  public interface ICalculator
  {
    double Addition(double operand1, double operand2);
    double Subtraction(double operand1, double operand2);
    double Multiplication(double operand1, double operand2);
    double Division(double operand1, double operand2);
  }
  public class CalculatorStub : ICalculator
  {
    public double Addition(double operand1, double operand2)
    {
      return 3;
    }
    public double Subtraction(double operand1, double operand2)
    {
      return -1;
    }
```

```
    public double Multiplication(double operand1, double operand2)
    {
      return 2;
    }
    public double Division(double operand1, double operand2)
    {
      return 0.5;
    }
  }
  public class Calculator : ICalculator
  {
    public double Addition(double operand1, double operand2)
    {
      return operand1 + operand2;
    }
    public double Subtraction(double operand1, double operand2)
    {
      return operand1 - operand2;
    }
    public double Multiplication(double operand1, double operand2)
    {
      return operand1 * operand2;
    }
    public double Division(double operand1, double operand2)
    {
      return operand1 / operand2;
    }
  }
}
```

در پیاده‌سازی جدید کلاس Calculator، کلاسی به نام CalculatorStub اضافه شده‌است که برای انجام آزمون واحد متدی که درون خود متدهای کلاس Calculator را فراخوانی می‌کند، ضروری است.

اکنون می‌توان متد MyMethod کلاس MyClass را تحت آزمون واحد قرار داد. بدین‌منظور مطابق آنچه برای متدهای کلاس Calculator گفته شد، عمل می‌کنیم. سپس در پوشه ProjectTests در قسمت Solution Explorer شاهد ایجاد کلاس دیگری با نام MyClassTests.cs خواهیم بود که محتویات آن به‌صورت زیر است.

```
using NUnit.Framework;

namespace Project.Tests
{
  [TestFixture()]
  public class MyClassTests
  {
    [Test()]
    public void MyMethodTest()
    {
      Assert.Fail();
    }
  }
```

```
}
```

این کلاس را به‌صورت زیر بازنویسی می‌کنیم.

```
using System;
using System.IO;
using NUnit.Framework;

namespace Project.Tests
{
   [TestFixture()]
   public class MyClassTests
   {
      [Test()]
      public void MyMethodTest()
      {
         var output = new StringWriter();
         Console.SetOut(output);
         var input = new StringReader("0");
         Console.SetIn(input);
         ReaderStub readerStub = new ReaderStub();
         CalculatorStub calculatorStub = new CalculatorStub();
         MyClass myClass = new MyClass(readerStub, calculatorStub);
         myClass.MyMethod();
         Assert.That(output.ToString(), Is.EqualTo("Result: 3"));
      }
   }
}
```

حال تمامی آزمون‌های نوشته‌شده را اجرا می‌کنیم. بدین‌منظور باید مراحل زیر طی شود.

Visual Studio ⇒ Test ⇒ Run ⇒ All Tests.

می‌توان نتایج آزمون واحد برای هریک از متدها را در بخش Test Explorer مشاهده کرد. برای مثال درمورد مثال‌های موجود در این گزارش نتایج به‌صورت زیر است.