Национальный Исследовательский Университет
«Московский Энергетический Институт»
Кафедра прикладной математики и искусственного интеллекта

Тема: Разработка модульного теста для кода на языке C# с применением MSUnit.

Студент: Ростовых Александра

Москва 2021

Цель работы

Научиться разрабатывать модульные тесты для кода на языке C# с применением библиотеки MSUnit. Разработать модульный тест с применением MSUnit.

1. Создать с помощью MS Visual Studio проект консольного приложения C# (Console Application).

Создали с помощью MS Visual Studio проект консольного приложения C# (Console Application).

2. Создать несколько интерфейсов, для классов, которые будут имитировать тестируемую логику. Создать классы-реализации этих интерфейсов.

#Интерфейс Persons.cs

```
namespace МКПЛаб3
{
   public interface Persons
   {
      public double IMT(double w, double height1);
      public double IMT();
      public int category(int age1, double w, double height1);
      public bool IsCorrect();
      public void FullName();
   }
}
```

#Класс Person.cs

```
namespace МКПЛаб3
{
   public class Person: Persons
   {
      public int age;
      public string name;
      public double weight, height;
      double def = 50;
      double defh = 165;
      public Person(string str, int age1=18,double w=50, double height1=165)
      {
         name = str;
         age = age1;
         weight = w;
         height = height1;
```

```
}
public Person()
{
   name = "Ростовых";
   age = 0;
   weight = 0;
   height = 0;
}
public double IMT(double w, double height1)
{
   if (w > 0 && height1>0)
   {
      return w/(height1*height1/10000);
   }
   else return 0;
}

public double IMT()
{
   return (weight > 0 && height > 0) ? (weight / (height * height / 10000)) : 0;

}
public int category( int age1, double w, double height1)
{
   if (age1>0)
   {
      if (w > 0 && height > 0)
         return (int)Math.Abs(age1 - IMT(w, height1))+1;
      else return (int)(IMT(def, defh)/5)+2;
   }
   else return -1;
}

public bool IsCorrect()
{
   return (name == "" || age < 0 || weight < 0 || height < 0) ? false : true;
}
public void FullName()
{
   if (this.name == "")
   {
      throw new PersonException("Name can't be empty");
   }
   else
   {
      this.name = "Ростовых Александра Дмитриевна";
   }

}
```

```
    }
}
```

#Интерфейс Students.cs

```
namespace МКПЛаб3
{
    public interface Students
    {
        public double averagescore(int Sum, int count);
        public void Newscore(int score);
        public int ID(int Sum, int count);
        string Name { get; }
        double Score { get; }
    }
}
```

#Класс Student.cs

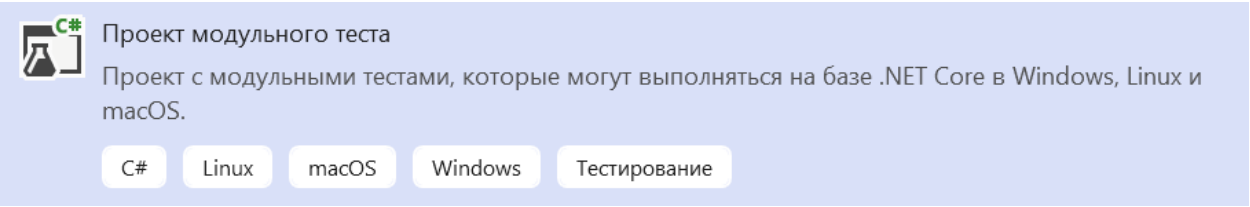
```
namespace МКПЛаб3
{
    public class Student: Students
    {
        public string name;
        public int score;
        public int Sum;
        public string Name => this.name;
        public double Score => this.score;
        public Student(string name, int score = 0)

        {
            this.name = name;
            this.score = score;
            this.Sum = score;
        }
        public void Newscore(int score)
        {
            this.score += score;
            this.Sum += score;
        }
        public double averagescore(int Sum, int count)
        {
            return (double)Sum / count;
        }

        public int ID(int Sum, int count)
        {
            return (int)(averagescore(Sum, count)/5+this.score*234-this.Sum);
        }
```

```
    }
}
```

#Класс PersonException.cs

```
namespace МКПЛаб3
{
    public class PersonException: Exception
    {
        public PersonException(string msg) : base(msg) { }
    }
}
```

3. Сгенерировать тестовый проект с модульными тестами.



4. Реализовать не менее пяти тестирующих функций. При разработке этих функций следует активно применять функции класса Assert.

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using МКПЛаб3;
using Moq;
    namespace TestProject1
    {
        [TestClass]
        public class UnitTest1
        {
            [TestMethod]
            public void TestMethod1()
            {
                Person person = new Person();
                Assert.AreEqual(0.0, person.IMT());
                Assert.IsTrue(person.IsCorrect());
                Assert.IsNotNull(person.name);
            }
        [TestMethod]
        public void TestMethod2()
        {
            Person person = new Person("", 21, 54, 172);
            Assert.AreEqual(25.0, person.IMT(100,200));
            Assert.IsFalse(person.IsCorrect());
            Assert.AreNotEqual(0.0, person.IMT());
```

```
}

[TestMethod]

public void TestMethod3()
{
   Person person = new Person("", 20);
   int age2 = 20;
   try
   {
      person.FullName();
   }
   catch (System.Exception e)
   {
      System.Console.WriteLine(e.Message);
   }
   Assert.IsFalse(age2 != person.age);
}
[TestMethod]
[ExpectedException(typeof(PersonException))]
public void TestMethod4()
{
   Person person = new Person("", 23, 55);
   person.FullName();
   Assert.AreEqual(1, person.category(18, 50, 170));
}
[TestMethod]
public void TestMethod5()
{
   Student student = new Student("Ростовых", 20);
   Assert.AreEqual(5.0, student.averagescore(student.Sum,4));
   Assert.IsNotNull(student.Name);
   Assert.IsNotNull(student.Score);
}
[TestMethod]
public void TestMethod6()
{
   int count = 9;
   Student student = new Student("Александра", 40);
   Assert.IsFalse(4.5 == student.averagescore(student.Sum, count));
   student.Newscore(5);
   count += 1;
   Assert.IsTrue(4.5 == student.averagescore(student.Sum, count));
   Assert.IsNotNull(student.Score);
}

[TestMethod]
public void TestMethod7()
{
```

```
    Mock<Students> mock = new Mock<Students>();
    mock.Setup(m => m.Name).Returns("Александра");
    Students m = mock.Object;
    Assert.IsNotNull(m.Name);
    Assert.IsTrue("Александра" == m.Name);
  }

  [TestMethod]
  public void TestMethod8()
  {
    int i = 5;
    Mock<Students> mock = new Mock<Students>();
    mock.Setup(m => m.Score).Returns(i);
    Students m = mock.Object;
    Assert.AreEqual(i, m.Score);
  }

  [TestMethod]
  public void TestMethod9()
  {
    Mock<Persons> mock = new Mock<Persons>();
    mock.Setup(m => m.IMT(50,167)).Returns(18);
    Persons m = mock.Object;
    Assert.AreEqual(18, m.IMT(50,167));
    Assert.AreNotEqual(17, m.IMT(50, 167));
  }
  [TestMethod]
  public void TestMethod10()
  {
    Mock<Persons> mock = new Mock<Persons>();
    mock.Setup(m => m.IsCorrect()).Returns(true);
    Persons m = mock.Object;
    Assert.IsTrue(m.IsCorrect());
  }


}
}
```

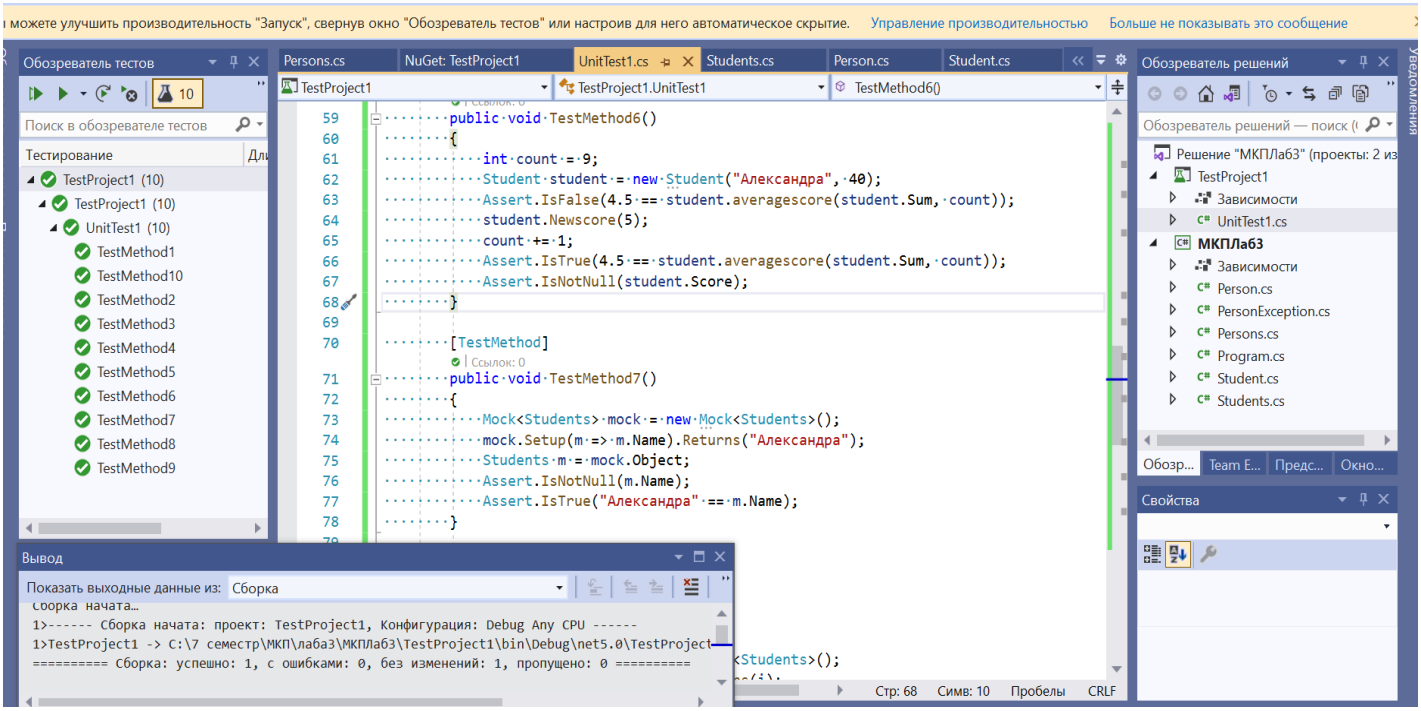
5. Разработать тестовый метод со спецификацией ожидаемых исключений.

```
  [TestMethod]
  [ExpectedException(typeof(PersonException))]
  public void TestMethod4()
  {
    Person person = new Person("", 23, 55);
    person.FullName();
    Assert.AreEqual(1, person.category(18, 50, 170));
  }
```

Так же были разработаны тестовые методы с использованием оболочки:

```
[TestMethod]
public void TestMethod7()
{
   Mock<Students> mock = new Mock<Students>();
   mock.Setup(m => m.Name).Returns("Александра");
   Students m = mock.Object;
   Assert.IsNotNull(m.Name);
   Assert.IsTrue("Александра" == m.Name);
}

[TestMethod]
public void TestMethod8()
{
  int i = 5;
   Mock<Students> mock = new Mock<Students>();
   mock.Setup(m => m.Score).Returns(i);
   Students m = mock.Object;
   Assert.AreEqual(i, m.Score);
}

[TestMethod]
public void TestMethod9()
{
   Mock<Persons> mock = new Mock<Persons>();
   mock.Setup(m => m.IMT(50,167)).Returns(18);
   Persons m = mock.Object;
   Assert.AreEqual(18, m.IMT(50,167));
   Assert.AreNotEqual(17, m.IMT(50, 167));
}
[TestMethod]
public void TestMethod10()
{
   Mock<Persons> mock = new Mock<Persons>();
   mock.Setup(m => m.IsCorrect()).Returns(true);
   Persons m = mock.Object;
   Assert.IsTrue(m.IsCorrect());
}
```

6. Запустить модульные тесты. Проанализировать результаты запуска



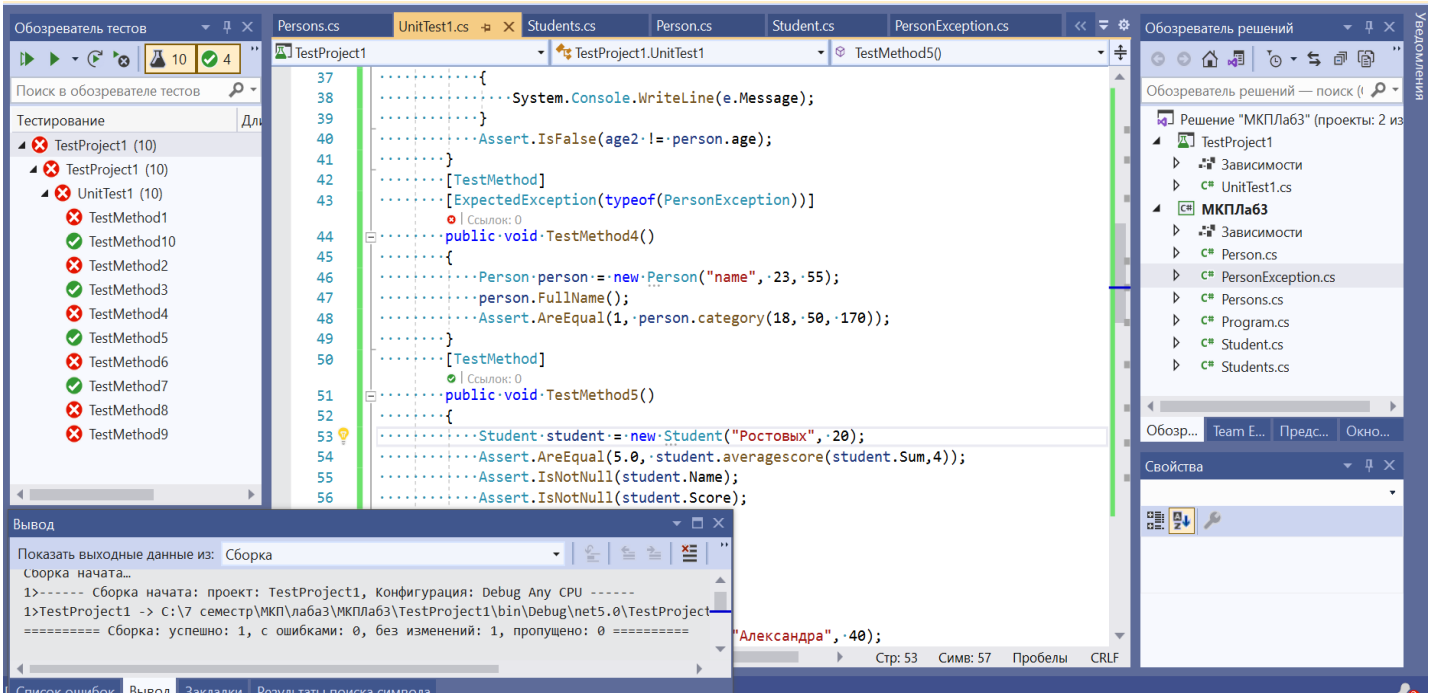
Все тесты успешно пройдены!

7. Внести в тестируемые классы изменения, приводящие к ошибкам.

Внесем некоторые изменения в тестируемые классы в методах 1,2,4,6,8,9:

```
[TestMethod]
        public void TestMethod1()
        {
            Person person = new Person();
            Assert.AreEqual(17.0, person.IMT());
            Assert.IsTrue(person.IsCorrect());
            Assert.IsNotNull(person.name);
        }
    [TestMethod]
    public void TestMethod2()
    {
        Person person = new Person("", 21, 54, 172);
        Assert.AreEqual(25.0, person.IMT(80,200));
        Assert.IsFalse(person.IsCorrect());
        Assert.AreNotEqual(0.0, person.IMT());
    }
[TestMethod]
public void TestMethod4()
    {
        Person person = new Person("name", 23, 55);
        person.FullName();
        Assert.AreEqual(1, person.category(18, 50, 170));
    }
[TestMethod]
    public void TestMethod6()
```

```
    {
        int count = 9;
        Student student = new Student("Александра", 40);
        Assert.IsFalse(4.5 == student.averagescore(student.Sum, count));
        student.Newscore(3);
        count += 1;
        Assert.IsTrue(4.5 == student.averagescore(student.Sum, count));
        Assert.IsNotNull(student.Score);
    }
[TestMethod]
    public void TestMethod8()
    {
      int i = 5;
        Mock<Students> mock = new Mock<Students>();
        mock.Setup(m => m.Score).Returns(i);
        Students m = mock.Object;
        Assert.AreEqual(0, m.Score);
    }

    [TestMethod]
    public void TestMethod9()
    {
        Mock<Persons> mock = new Mock<Persons>();
        mock.Setup(m => m.IMT(50,167)).Returns(18);
        Persons m = mock.Object;
        Assert.AreEqual(17, m.IMT(50,167));
        Assert.AreNotEqual(18, m.IMT(50, 167));
    }
```



Эти же тесты оказались с ошибками.