

## **Лабораторная работа: классы C# .**

### **Вопросы для обсуждения**

- Описание класса
- Простые свойства. Индексаторы.
- Метод для чтения.
- Метод для записи.
- Связь свойства с полем.
- Свойства с индексом.
- Модульное тестирование.

Разработка класса Матрицы.

## **Лабораторная работа. Класс для работы с матрицами.**

Тема: Классы.

*Цель: Сформировать практические навыки описания и использования классов и свойств для обработки данных хранимых в объектах классов.*

### **Задание**

Разработайте класс Матрица (Matrix) для операций матричной алгебры и протестируйте методы этого класса в режиме консольного приложения.

### **Спецификация типа данных Матрица**

**ADT Matrix**

### **Данные**

Матрица (тип Matrix) - это двумерная матрица со значениями целого типа. Объект типа Матрица - изменяемый.

### **Операции**

<b>Конструктор (Matrix)</b>	
Вход:	Число строк $i$ и столбцов $j$ .
Предусловия:	Число строк и столбцов должно быть больше 0.
Процесс:	Создаёт объект типа Matrix с заданным числом строк и столбцов. Заносит число строк и столбцов в соответствующие свойства I и J.
<b>Сложить (operator+)</b>	
Вход:	(b) – объект тип Matrix.
Предусловия:	Число строк и столбцов в суммируемых матрицах должны совпадать
Процесс:	Создаёт новый объект типа Matrix, элементы которого, получены путём сложения элементов объектов this и b с одинаковыми индексами.
Выход:	Объект типа Matrix.
Постусловия:	Нет.
<b>Вычитать (operator-)</b>	
Вход:	(b) – объект тип Matrix.
Предусловия:	Число строк и столбцов в матрицах, участвующих в вычитании, должны совпадать.
Процесс:	Создаёт новый объект типа Matrix, элементы которого, получены путём вычитания элементов объектов this и b с одинаковыми индексами.

Выход:	Объект типа Matrix.
Постусловия:	Нет.
<b>Умножить (operator*)</b>	
Вход:	(b) – объект типа Matrix.
Предусловия:	Матрицы, участвующие в умножении, должны быть согласованы для этой операции по числу строк и столбцов.
Процесс:	Создаёт новый объект типа Matrix, элементы которого, получены путём умножения элементов объектов this и b в соответствии с правилами перемножения матриц.
Выход:	Объект типа Matrix.
Постусловия:	Нет.
<b>Равно (operator==)</b>	
Вход:	(b) – объект типа Matrix.
Предусловия:	Число строк и столбцов в матрицах, участвующих в вычитании, должны совпадать.
Процесс:	Возвращает значение true, если элементы объектов this и b в на одинаковых позициях равны.
Выход:	Значение типа bool.
Постусловия:	Нет.
<b>Транспонировать (Transp)</b>	
Вход:	Нет.
Предусловия:	Матрица, подвергаемая транспонированию, должна иметь одинаковое число строк и столбцов.
Процесс:	Создаёт новый объект типа Matrix, элементы которого, получены путём транспонирования элементов объекта this.
Выход:	Объект типа Matrix.
Постусловия:	Нет.
<b>Минимальный элемент (Min)</b>	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Отыскивает и возвращает минимальный среди элементов объекта this.

Выход:	Значение типа int.
Постусловия:	Нет.
<b>Преобразовать В строку (ToString)</b>	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Преобразует элементы матрицы this в строковое представление построчно. Напрмер: {{1,2,3},{4,5,6},{7,8,9}}
Выход:	Нет.
Постусловия:	Нет.
<b>Взять элемент с индексами i,j (this [i,j])</b>	
Вход:	Значения i, j типа int.
Предусловия:	Значения i, j должны находиться в допустимых диапазонах.
Процесс:	Возвращает элемент матрицы с индексами <i>i,j</i> .
Выход:	Значение типа int.
Постусловия:	Нет.
<b>Записать элемент с индексами i,j (this [i,j])</b>	
Вход:	Значения i, j типа int, n - новое значение элемента типа int.
Предусловия:	Значения i, j должны находиться в допустимых диапазонах.
Процесс:	Заменяет элемент матрицы с индексами i,j на значение n.
Выход:	Нет.
Постусловия:	Элемент матрицы с индексами i,j получает значение n.

**end Matrix**

#### **Рекомендации к выполнению**

1. Тип данных реализовать, используя класс C#.
2. Матрицу храните в поле типа двумерный массив целого типа.
3. Для доступа к элементам матрицы используйте индексатор.
4. Для доступа к числу строк и столбцов используйте свойство (property).
5. Тип данных реализовать в отдельном файле Matrix.
6. Для тестирования используйте модульные тесты.

Ниже приведён пример описания класса Matrix:

//-----

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplicationMatrix
{
    public class MyException: Exception
    {
        public MyException(string s): base(s)
        { }
    }
    public class Matrix
    {
        int[, ] m;
        //Свойство для работы с числом строк.
        public int I { get; set; }
        //Свойство для работы с числом столбцов.
        public int J { get; set; }
        //Конструктор.
        public Matrix(int i, int j)
        {
            if ( i <= 0 ) throw new
MyException(string.Format("недопустимое значение i = {0}", i));
            if ( j <= 0 ) throw new
MyException(string.Format("недопустимое значение j = {0}", j));
            I = i;
            J = j;
            m = new int[i, j];
        }
        //Индексатор для доступа к значениям компонентов матрицы.
        public int this[int i, int j]
        {
            get
            {
                if ( i < 0 | i > I - 1 ) throw new
MyException(string.Format("неверное значение i = {0}", i));
                if ( j < 0 | j > J - 1 ) throw new
MyException(string.Format("неверное значение j = {0}", j));
                return m[i, j];
            }
            set
            {
                if (i < 0 | i > I - 1) throw new
MyException(string.Format("неверное значение i = {0}", i));
                if (j < 0 | j > J - 1) throw new
MyException(string.Format("неверное значение j = {0}", j));
                m[i, j] = value;
            }
        }
        //Сложение матриц.
        public static Matrix operator+(Matrix a, Matrix b)
        {
            Matrix c = new Matrix(a.I, a.J);

```

```

        for (int i = 0; i < a.I; i++)
            for (int j = 0; j < a.J; j++)
            {
                c[i,j] = a.m[i, j] + b.m[i,j];
            }
        return c;
    }
    public static bool operator ==(Matrix a, Matrix b)
    {
        bool q = true;
        for (int i = 0; i < a.I; i++)
            for (int j = 0; j < a.J; j++)
            {
                if (a[i, j] != b[i, j])
                {
                    q = false; break;
                }
            }
        return q;
    }
    public static bool operator !=(Matrix a, Matrix b)
    {
        return !(a==b);
    }
    //Вывод значений компонентов на консоль.
    public void Show()
    {
        for (int i = 0; i < I; i++)
        {
            for (int j = 0; j < J; j++)
            {
                Console.Write("\t" + this[i,j] );
            }
            Console.WriteLine();
        }
        Console.WriteLine();
    }
    public override bool Equals(object obj)
    {
        return (this as Matrix) == (obj as Matrix);
    }
}

```

Текст файла ConsoleApplicationMatrix, содержащего консольное приложение ConsoleApplicationMatrix приведено ниже:

```

//-----
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplicationMatrix
{

```

```

class Program
{
    static void Main(string[] args)
    {
        try
        {
            //Создаём матрицу a.
            Matrix a = new Matrix(3, 3);
            //Создаём матрицу b.
            Matrix b = new Matrix(3, 3);
            //Объявляем матрицу c.
            Matrix c;
            //Заполняем матрицу a.
            for (int i = 0; i < a.I; i++)
            {
                for (int j = 0; j < a.J; j++)
                {
                    a[i, j] = a.J * i + j;
                }
            }
            //Выводим матрицу a.
            a.Show();
            //Заполняем матрицу b.
            for (int i = 0; i < a.I; i++)
            {
                for (int j = 0; j < a.J; j++)
                {
                    b[i, j] = a.J * i + j + 1;
                }
            }
            //Выводим матрицу a.
            b.Show();
            //Складываем матрицы a и b.
            c = a + b;
            //Выводим матрицу c.
            c.Show();
        }
        catch (MyException e)
        {
            Console.WriteLine(e.Message);
        }
    }
}

```

Запустив это приложение, мы увидим на экране:

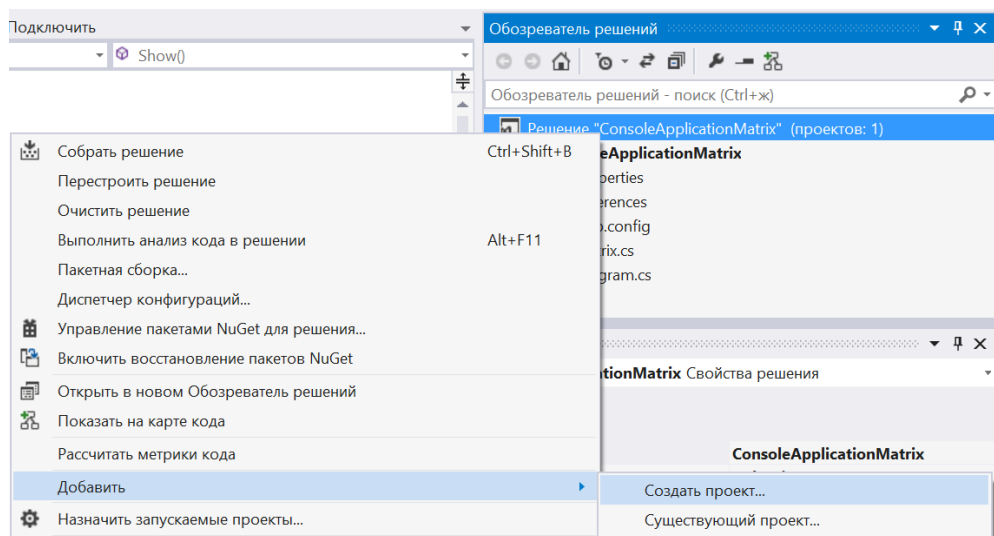
0	1	2
3	4	5
6	7	8
1	2	3
4	5	6
7	8	9
1	3	5
7	9	11
13	15	17

Для продолжения нажмите любую клавишу . . .

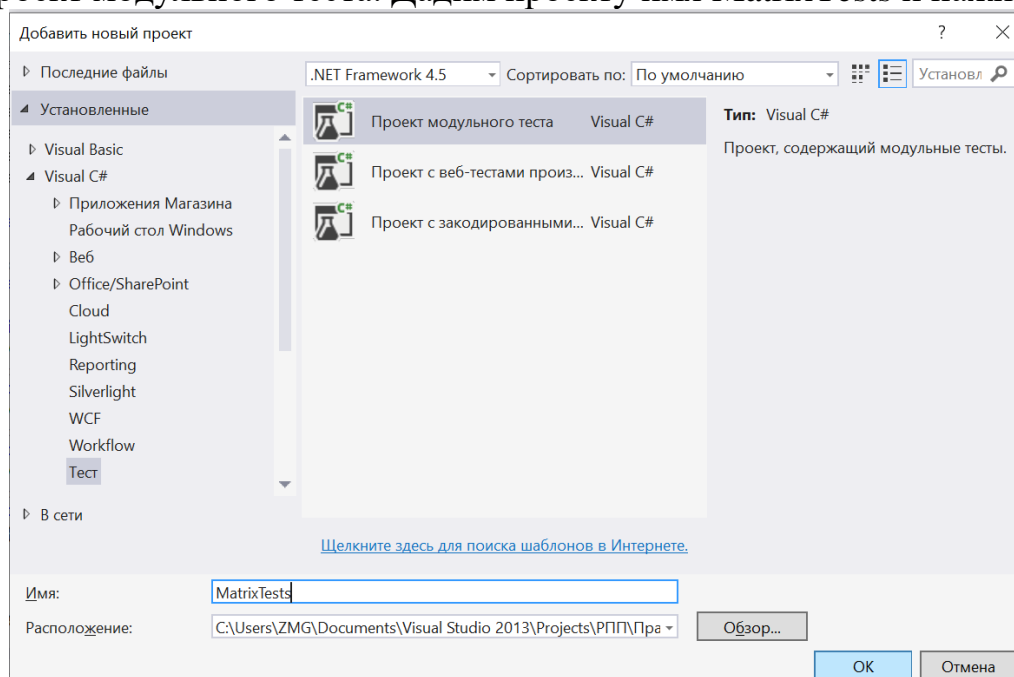
## Модульное тестирование класса средствами Visual Studio.

Для тестирования классов в Visual Studio имеется проект модульного теста. С помощью модульного теста вы можете протестировать все методы класса. Для тестирования классов вашего проекта вам необходимо добавить в решение, в котором находится тестируемый проект, добавить проект модульного теста. Затем необходимо сделать классы проекта доступными в проекте модульного теста.

Добавим в наше решение проект модульного теста MatrixTests. В окне **Обозреватель решений** необходимо по правой клавише мыши во всплывающих меню выбрать команды **Добавить > Создать проект**.

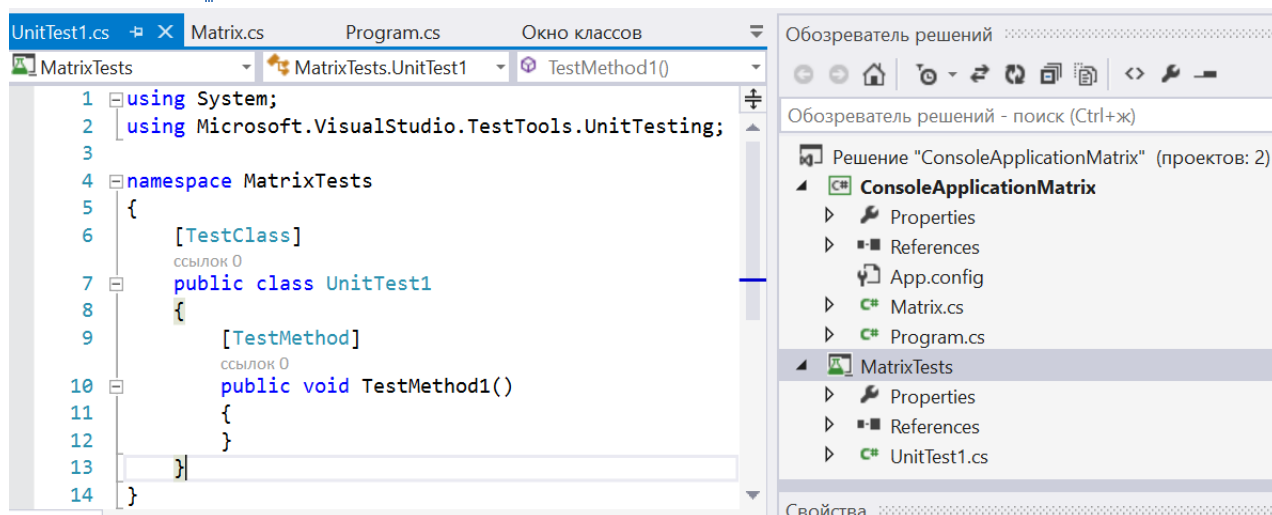


В появившемся окне **Добавить новый проект** выберите **Visual C# > Тест > Проект модульного теста**. Дадим проекту имя **MatrixTests** и нажимаем **ОК**.



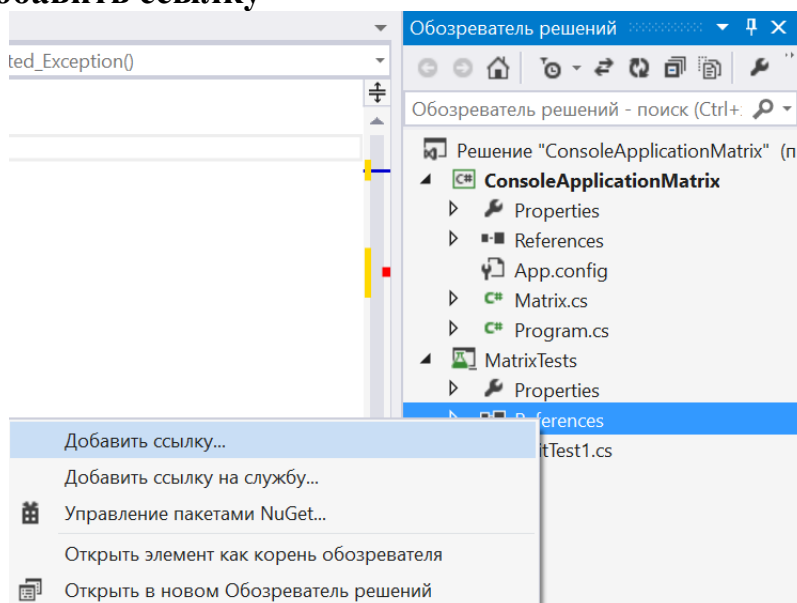
Открывается окно редактора с заготовкой класса модульного теста **UnitTest1**, в котором имеется один тестовый метод **TestMethod1**.



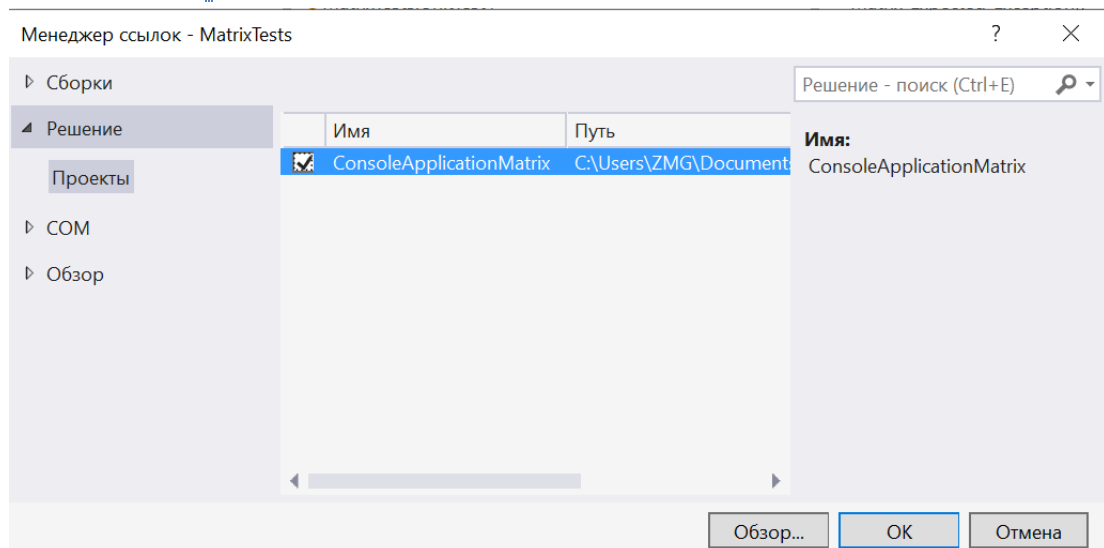


Тестовый класс имеет обязательный атрибут [TestClass], который предшествует заголовку класса. Тестовый метод также имеет обязательный атрибут [TestMethod], который предшествует заголовку метода. Операторы необходимые для выполнения тестов необходимо помещать в тестовые методы. В тестовый класс вы можете добавить произвольное число тестовых методов.

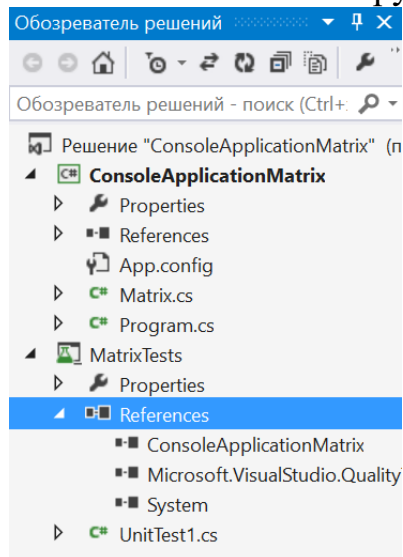
Добавим ссылку на тестируемый проект ConsoleApplicationMatrix с помощью окна **Обозреватель решений**. Выполните команды **References > Добавить ссылку**



В появившемся окне **Менеджер ссылок** в Решение>Проекты поставьте галочку напротив имени тестируемого проекта.



После этого в окне Обозреватель решений в проекте модульного теста появится ссылка на тестируемый проект.



Изменим уровень видимости для классов Matrix, MyException на public

```
public class MyException: Exception
public class Matrix
```

Поменяем имя файла содержащего тестовый класс и имя тестового класса на MatrixTests.

Добавим предложение using в файл модульного теста

```
using ConsoleApplicationMatrix;
```

Поменяем имя метода Method1 на Matrix\_Expected\_MyException\_i. Потому что в этом методе мы протестируем возбуждение исключительной ситуации в конструкторе при недопустимом значении число строк i в матрице. Добавим ещё тестовый метод для тестирования исключения при недопустимом значении числа столбцов j в матрице Matrix\_Expected\_MyException\_j. Тогда текст файла модульного теста примет следующий вид:

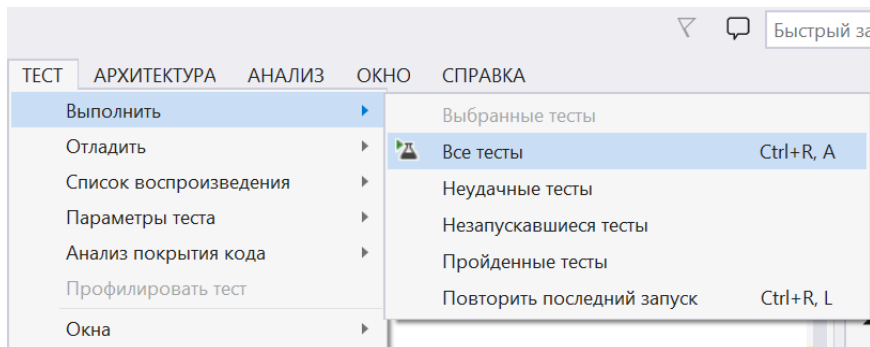
```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using ConsoleApplicationMatrix;
namespace MatrixTests
```

```

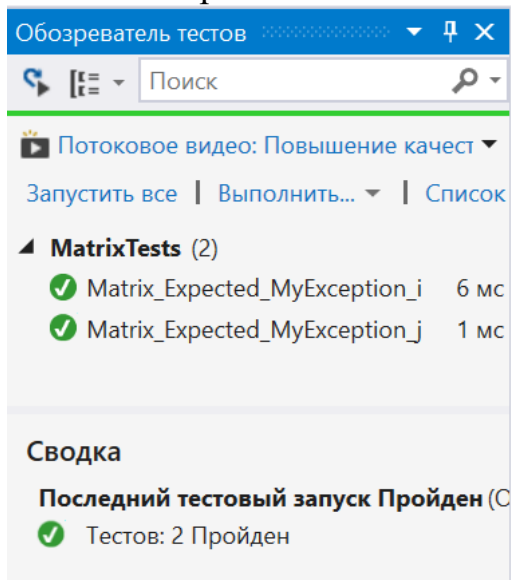
{
    [TestClass]
    public class MatrixTests
    {
        [TestMethod]
        [ExpectedException(typeof(MyException))] //Тип ожидаемого исключения.
        public void Matrix_Expected_MyException_i()
        {
            Matrix a = new Matrix(0, 2);
        }
        [TestMethod]
        [ExpectedException(typeof(MyException))] //Тип ожидаемого исключения.
        public void Matrix_Expected_MyException_j()
        {
            Matrix a = new Matrix(2, -1);
        }
    }
}

```

Запустим тесты на выполнение, выполнив команды **Выполнить > Все тесты**



В окне **Обозреватель тестов** получим результат



Оба теста успешно пройдены.

Добавим в тестовый класс тестовые методы для тестирования работы индексатора, операции равно и операции суммирования. Полученный тестовый класс представлен ниже

```

using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using ConsoleApplicationMatrix;
namespace MatrixTests

```

```

{
    [TestClass]
    public class MatrixTests
    {
        [TestMethod]
        [ExpectedException(typeof(MyException))]
        public void Matrix_Expected_MyException_i()
        {
            //act (выполнить)
            Matrix a = new Matrix(0, 2);
        }
        [TestMethod]
        [ExpectedException(typeof(MyException))]
        public void Matrix_Expected_MyException_j()
        {
            //act (выполнить)
            Matrix a = new Matrix(2, -1);
        }
        [TestMethod]
        [ExpectedException(typeof(MyException))]
        public void this_Expected_MyException_set_j()
        {
            //act (выполнить)
            Matrix a = new Matrix(2, 2);
            a[1, 3] = 2;
        }
        [TestMethod]
        [ExpectedException(typeof(MyException))]
        public void this_Expected_MyException_get_i()
        {
            //act (выполнить)
            Matrix a = new Matrix(2, 2);
            int r = a[3, 1];
        }
        [TestMethod]
        public void Equal()
        {
            //arrange(обеспечить)
            Matrix a = new Matrix(2, 2);
            a[0, 0] = 1; a[0, 1] = 1; a[1, 0] = 1; a[1, 1] = 1;
            Matrix b = new Matrix(2, 2);
            b[0, 0] = 1; b[0, 1] = 1; b[1, 0] = 1; b[1, 1] = 1;
            //act (выполнить)
            //bool r = a == b;
            //assert(доказать)
            //Assert.IsTrue(r);
            Assert.AreEqual(a, b);
        }
        [TestMethod]
        public void Summa()
        {
            //arrange(обеспечить)
            Matrix a = new Matrix(2, 2);
            a[0, 0] = 1; a[0, 1] = 1; a[1, 0] = 1; a[1, 1] = 1;
            Matrix b = new Matrix(2, 2);
            b[0, 0] = 2; b[0, 1] = 2; b[1, 0] = 2; b[1, 1] = 2;
            Matrix expected = new Matrix(2, 2);
            expected[0, 0] = 3; expected[0, 1] = 3;
            expected[1, 0] = 3; expected[1, 1] = 3;
            Matrix actual = new Matrix(2, 2);
            //act (выполнить)

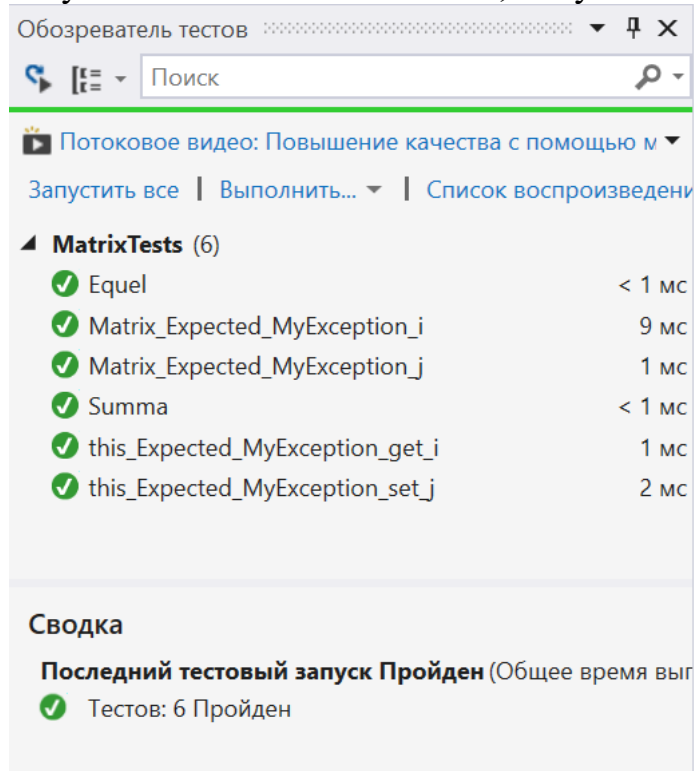
```

```

        actual = a + b;
        //assert(доказать)
        Assert.IsTrue(actual == expected); //Оракул
    }
}
}

```

Запустив тесты на выполнение, получим следующий результат



#### Указания к выполнению

1. Создайте проект – Консольное приложение под именем ConsoleApplicationMatrix. В его состав включите класс Matrix.
2. Добавьте в решение проект Модульный тест.
3. Реализуйте в соответствии с заданием класс Matrix. Протестируйте Matrix с помощью класса модульного теста, добавляя в него необходимые тестовые методы.

Таблица 1. Тестовый набор.

Тестовый набор					
Номер теста	Исходные данные				Ожидаемый результат
1					
2					

#### Содержание отчета

1. Задание.
2. Исходные тексты приложения.
3. Исполняемый файл приложения.
4. Результат тестирования методов класса Matrix.

### **Контрольные вопросы**

1. Синтаксис описания свойства?
2. Сигнатура метода для чтения свойства?
3. Сигнатура метода для записи свойства?
4. Синтаксис индексаторов?
5. Назначение свойства?
6. Использование свойства?