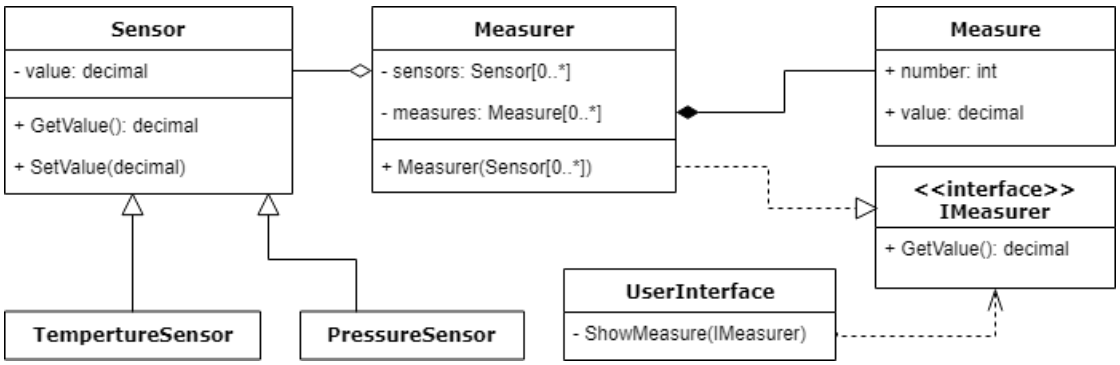
ДЕНЬ 7. СПЕЦИФИКАЦИИ 1

7.1. Создать декларацию классов C# согласно заданной UML-диаграмме классов

Создать код декларации классов (без их реализации) на языке C# по заданной UML-диаграмме классов

Пример

Диаграмма классов:

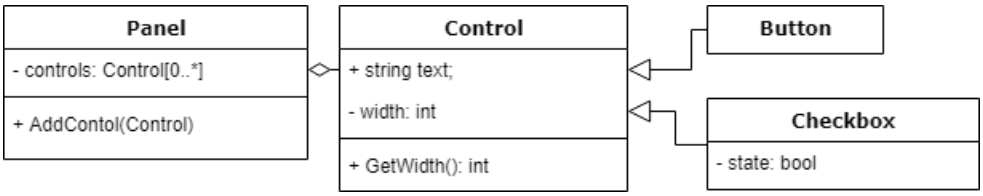
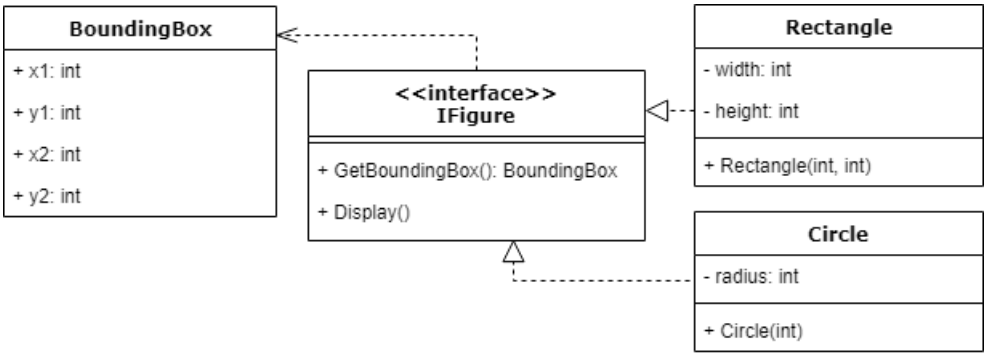
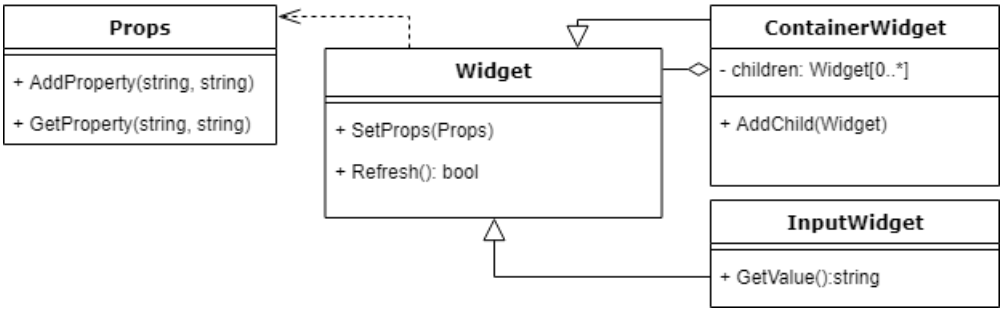
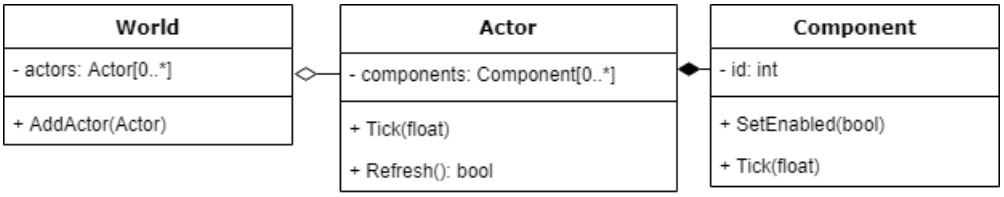
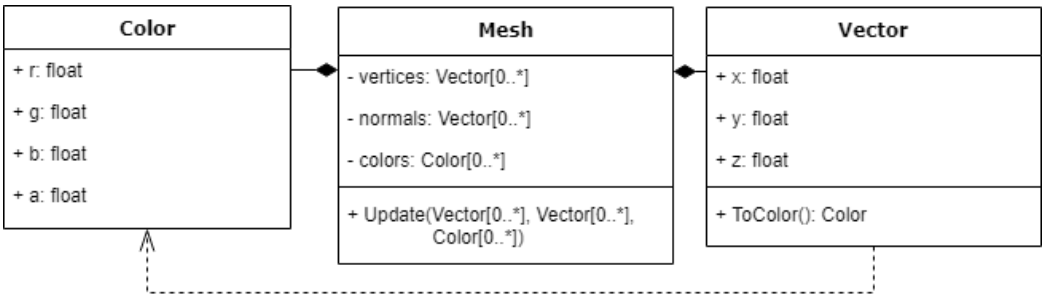


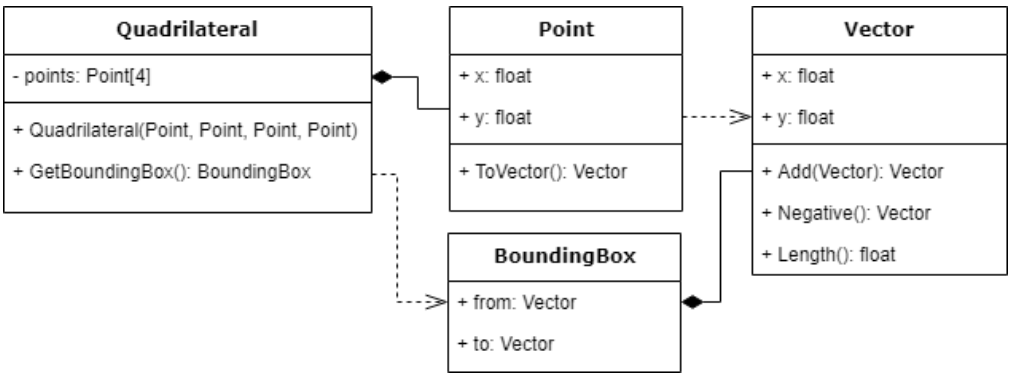
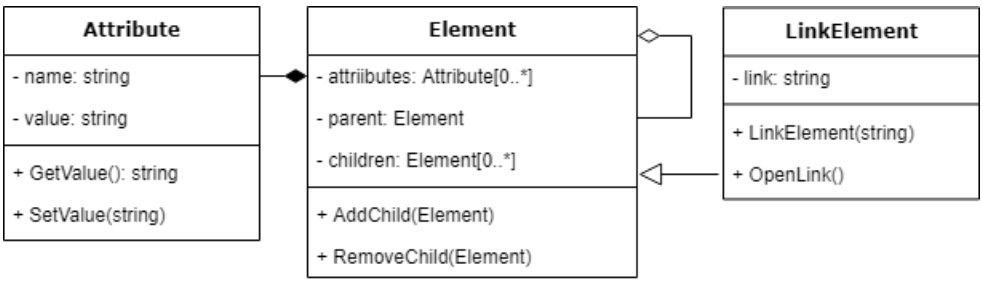
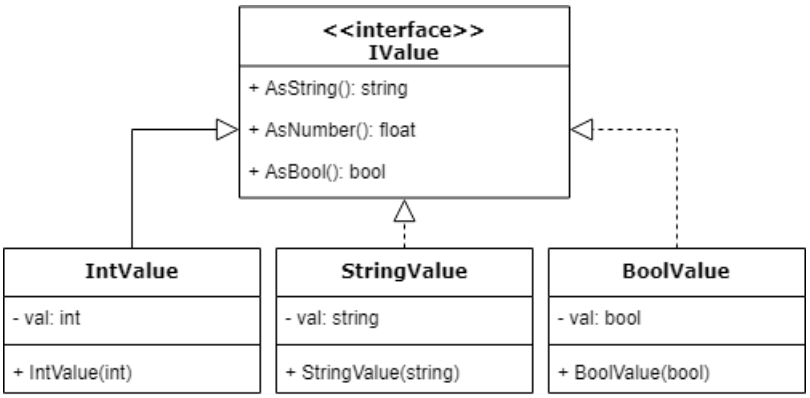
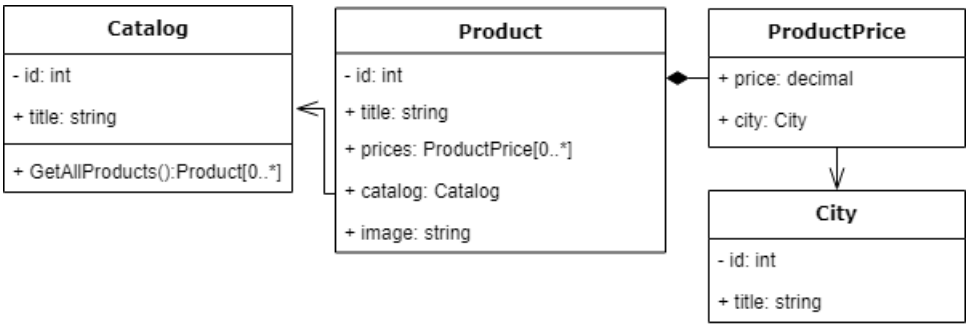
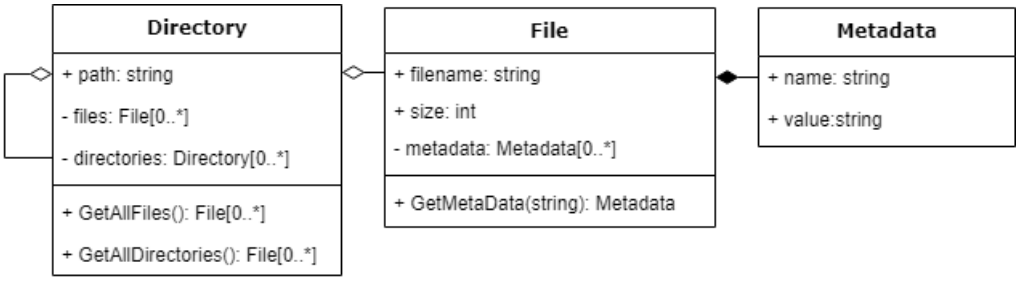
Код декларации классов:

```
public class Sensor
{
    private decimal value;

    public decimal GetValue()
    {
        throw new NotImplementedException();
    }

    public void SetValue(decimal val)
    {
        throw new NotImplementedException();
    }
}

public class TempertureSensor: Sensor
{

}

public class PressureSensor: Sensor
{

}

public class Measure
{
    public int number;
    public decimal value;
}
```

```
public interface IMeasurer
{
    decimal GetValue();
}

public class Measurer: IMeasurer
{
    private List<Sensor> sensors;
    private List<Measure> measures;

    public Measurer(List<Sensor> sens)
    {
        throw new NotImplementedException();
    }

    public decimal GetValue()
    {
        throw new NotImplementedException();
    }
}

public class UserInterface
{
    private void ShowMeasure(IMeasurer)
    {
        throw new NotImplementedException();
    }
}
```

Варианты

№	Диаграмма
1	
2	
3	
4	
5	

6	
7	
8	
9	
10	
11	см. вариант 1
12	см. вариант 2

13	см. вариант 3
14	см. вариант 4
15	см. вариант 5
16	см. вариант 6
17	см. вариант 7
18	см. вариант 8
19	см. вариант 9
20	см. вариант 10

7.2. Разработать модульный тест для функции проверки правильности ввода данных, возвращающей код ошибки

Необходимо разработать функцию проверки правильности ввода данных для заданной записи и модульный тест, ее проверяющий. Функция проверки правильности ввода должна принимать на вход объект записи и возвращать, либо 0, если ошибок нет, либо номер поля с ошибкой (начиная с 1)

Решение должно состоять из двух проектов:

- *Библиотека классов (.NET Framework)*, в которой должен быть класс с функцией проверки правильности ввода данных и класс, представляющий собой запись
- *Проект модульного теста (.NET Framework),* в котором должен быть реализован модульный тест.

Модульный тест должен проверить, как положительный исход функции, так и все варианты отрицательного исхода.

Варианты

№	Поля записи и их проверка		
	Поле 1	**Поле 2**	**Поле 3**
1	Фамилия (строка): не меньше 3 букв	Должность (строка): только русские буквы	Оклад в руб (целое число): больше 0
2	Название цеха (строка): только буквы и цифры	План выпуска деталей (целое число): не больше 1000	Фактический выпуск деталей (целое число): не меньше 0
3	Город (строка): не больше 255 символов	Улица (строка): только русские буквы, знак дефиса, пробела и цифры	Номер дома (число): больше 0
4	Фамилия (строка): не меньше 3 букв	Рост (вещественное число): не больше 300	Вес (вещественное число): больше 30
5	Адрес отправления (строка): не меньше 30 символов	Адрес доставки (строка): не равен адресу отправления	Вес (вещественное число): не больше 90
6	Автомобильный номер (строка): только цифры и буквы А, В, Е, К, М, Н, О, Р, С, Т, У, Х	Год выпуска (целое число): от 1980 до текущего года включительно	Пробег в км (целое число): больше или равно 0

7	Название (строка): не больше 255 символов	Число сезонов (целое число): больше или равно одному	Год выпуска первого сезона (целое число): от 2000 до текущего года включительно
8	Фамилия (строка): не меньше 3 букв	Номер группы (строка): только цифры, знак дефиса и русские буквы	Номер в группе (целое число): больше 0
9	Фамилия (строка): не меньше 3 букв	Год поступления (целое число): не меньше 1952 и не больше текущего года	Средний балл (вещественное число): от 0 до 5 включительно
10	Фамилия (строка): не меньше 3 букв	Оценка за теорию (целое число): не меньше 3	Оценка за практику (целое число): не больше 5
11	Номер заказа (строка): только латинские буквы и цифры	Описание (строка): не меньше 200 символов	Сумма заказа (целое число): не меньше 0
12	Дисциплина (строка): только русские буквы, знаки дефиса и пробела	номер курса (целое число): от 0 до 6 включительно	количество часов (целое число): от 0 до 9999 включительно
13	Номер телефона (строка): только цифры, скобки, знаки тире, плюса и пробела	Имя оператора (строка): не меньше 3 символов	баланс в копейках (целое число): не меньше -9999999
14	Название товара (строка): не меньше 10 символов	Количество на складе (целое число): от 0 до 999 включительно	Количество зарезервированных (целое число): не меньше 0
15	Название материала (строка): не меньше 10 символов	Объем (вещественное число): не больше 1000	Вес (вещественное число): не меньше 0
16	Тема письма (строка): не больше 1024 символов	Адресат (строка): только латинские буквы, знак «@», точка, тире и цифры	Число слов (целое число): не меньше 1
17	Адрес сайта (строка): только латинские буквы, точка, тире и цифры	Число посетителей (целое число): не меньше числа уникальных посетителей	Число уникальных посетителей (целое число): не меньше 0
18	Производитель (строка): не меньше 10 символов	Объем выпуска (вещественное число): не меньше 0	Средняя цена (вещественное число): от 1 до 99999 включительно
19	Компания (строка), не меньше 10 символов	Сумма поступлений в млн. руб. (вещественное число): не меньше 0 и не больше 1000000	Сумма списаний в млн. руб. (вещественное число): не меньше 0
20	Фамилия (строка): не меньше 3 букв	Число ролей (целое число): не меньше 0 и не больше 9999	Гонорар в млн. руб. (вещественное число): не меньше 0

7.3. Разработать модульный тест для функции проверки правильности ввода данных, бросающей пользовательское исключение в случае ошибки

Необходимо разработать функцию проверки правильности ввода данных для заданной записи и модульный тест, ее проверяющий. Функция проверки правильности ввода должна принимать на вход объект записи. Если функция обнаружила ошибку в данных, она должна бросить **исключение вами созданного типа**, который помимо текста ошибки, должен содержать номер поля (начиная с 1), в котором обнаружена ошибка.

Для создания собственного типа исключения необходимо создать класс, который наследуется от класса *Exception*. Добавьте в него дополнительное свойство – номер поля в записи.

Решение должно состоять из двух проектов:

- *Библиотека классов (.NET Framework)*, в которой должен быть класс с функцией проверки правильности ввода данных, класс, представляющий собой запись, и класс собственного типа исключения
- *Проект модульного теста (.NET Framework)*, в котором должен быть реализован модульный тест.

Модульный тест должен проверить, как положительный исход функции, так и все варианты отрицательного исхода. В случае отрицательного исхода тест должен проверить как тип брошенного исключения (он должен быть вами созданным), так и номер поля.

Варианты

№	Поля записи и их проверка		
	Поле 1	**Поле 2**	**Поле 3**
1	Компания (строка), не меньше 10 символов	Сумма поступлений в млн. руб. (вещественное число): не меньше 0 и не больше 1000000	Сумма списаний в млн. руб. (вещественное число): не меньше 0
2	Номер телефона (строка): только цифры, скобки, знаки тире, плюса и пробела	Имя оператора (строка): не меньше 3 символов	баланс в копейках (целое число): не меньше -9999999
3	Фамилия (строка): не меньше 3 букв	Номер группы (строка): только цифры, знак дефиса и русские буквы	Номер в группе (целое число): больше 0
4	Номер заказа (строка): только латинские буквы и цифры	Описание (строка): не меньше 200 символов	Сумма заказа (целое число): не меньше 0
5	Название цеха (строка): только буквы и цифры	План выпуска деталей (целое число): не больше 1000	Фактический выпуск деталей (целое число): не меньше 0
6	Название товара (строка): не меньше 10 символов	Количество на складе (целое число): от 0 до 999 включительно	Количество зарезервированных (целое число): не меньше 0
7	Фамилия (строка): не меньше 3 букв	Число ролей (целое число): не меньше 0 и не больше 9999	Гонорар в млн. руб. (вещественное число): не меньше 0
8	Название материала (строка): не меньше 10 символов	Объем (вещественное число): не больше 1000	Вес (вещественное число): не меньше 0
9	Адрес сайта (строка): только латинские буквы, точка, тире и цифры	Число посетителей (целое число): не меньше числа уникальных посетителей	Число уникальных посетителей (целое число): не меньше 0
10	Автомобильный номер (строка): только цифры и буквы А, В, Е, К, М, Н, О, Р, С, Т, У, Х	Год выпуска (целое число): от 1980 до текущего года включительно	Пробег в км (целое число): больше или равно 0
11	Производитель (строка): не меньше 10 символов	Объем выпуска (вещественное число): не меньше 0	Средняя цена (вещественное число): от 1 до 99999 включительно

12	Фамилия (строка): не меньше 3 букв	Должность (строка): только русские буквы	Оклад в руб (целое число): больше 0
13	Фамилия (строка): не меньше 3 букв	Оценка за теорию (целое число): не меньше 3	Оценка за практику (целое число): не больше 5
14	Название (строка): не больше 255 символов	Число сезонов (целое число): больше или равно одному	Год выпуска первого сезона (целое число): от 2000 до текущего года включительно
15	Тема письма (строка): не больше 1024 символов	Адресат (строка): только латинские буквы, знак «@», точка, тире и цифры	Число слов (целое число): не меньше 1
16	Город (строка): не больше 255 символов	Улица (строка): только русские буквы, знак дефиса, пробела и цифры	Номер дома (число): больше 0
17	Дисциплина (строка): только русские буквы, знаки дефиса и пробела	номер курса (целое число): от 0 до 6 включительно	количество часов (целое число): от 0 до 9999 включительно
18	Фамилия (строка): не меньше 3 букв	Год поступления (целое число): не меньше 1952 и не больше текущего года	Средний балл (вещественное число): от 0 до 5 включительно
19	Фамилия (строка): не меньше 3 букв	Рост (вещественное число): не больше 300	Вес (вещественное число): больше 30
20	Адрес отправления (строка): не меньше 30 символов	Адрес доставки (строка): не равен адресу отправления	Вес (вещественное число): не больше 90

7.4. Создать метод, проходящий заданные модульные тесты

Необходимо реализовать метод *Calculate* для обработки массива целых чисел, который проходит заданные модульные тесты. Объявление метода должно выглядеть следующим образом:

```
public class MyTask
{
    static public int Calculate(int[] numbers)
    {
        // Тело метода
    }
}
```

Пример

Заданные тесты:

```
[TestClass]
public class UnitTest
{
    [TestMethod]
    public void CheckZero()
    {
        int[] input = new int[0];
        Assert.AreEqual(0, MyTask.Calculate(input);
    }
    [TestMethod]
    public void CheckSum()
    {
        int[] input = new int[] { 4, 8, 15, 16, 23, 42};
        Assert.AreEqual(108, MyTask.Calculate(input));
    }
}
```

Решение:

```
public class MyTask
{
    static public int Calculate(int[] numbers)
    {
        int sum = 0;
        foreach (int n in numbers) sum += n;
        return sum;
    }
}
```

Варианты

№	Тесты
1	[TestMethod] public void CheckMax() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4}; Assert.AreEqual(8, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(6, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(25, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMax() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(7, 10); int p = r.Next(5); int[] input = new int[] { r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7) }; input[p] = v; Assert.AreEqual(v, MyTask.Calculate(input)); } } [TestMethod] public void CheckZero() { int[] input = new int[0]; Assert.AreEqual(0, MyTask.Calculate(input)); }
2	[TestMethod] public void CheckMaxIndex() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(2, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(3, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(1, MyTask.Calculate(input3)); } [TestMethod]

	public void CheckRandMaxIndex() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(7, 10); int p = r.Next(5); int[] input = new int[] { r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7) }; input[p] = v; Assert.AreEqual(p, MyTask.Calculate(input)); } } [TestMethod] public void CheckMinusOne() { int[] input = new int[0]; Assert.AreEqual(-1, MyTask.Calculate(input)); }
3	[TestMethod] public void CheckAbsMax() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(8, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(7, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(32, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMax() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(7, 10); int p = r.Next(5); int[] input = new int[] { r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7) }; input[p] = v; Assert.AreEqual(v, MyTask.Calculate(input)); } } [TestMethod] public void CheckException() { int[] input = new int[0]; Assert.ThrowsException<Exception>(() => MyTask.Calculate(input)); }
4	[TestMethod] public void CheckMin() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(1, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(-7, MyTask.Calculate(input2));

	`` int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(-32, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMin() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(-5, 7); int p = r.Next(5); int[] input = new int[] { r.Next(7, 10), r.Next(7, 10), r.Next(7, 10), r.Next(7, 10), r.Next(7, 10) }; input[p] = v; Assert.AreEqual(v, MyTask.Calculate(input)); } } [TestMethod] public void CheckZero() { int[] input = new int[0]; Assert.AreEqual(0, MyTask.Calculate(input)); } ```
5	``` [TestMethod] public void CheckMinIndex() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(3, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(0, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(3, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMinIndex() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(-5, 7); int p = r.Next(5); int[] input = new int[] { r.Next(7, 10), r.Next(7, 10), r.Next(7, 10), r.Next(7, 10), r.Next(7, 10) }; input[p] = v; Assert.AreEqual(p, MyTask.Calculate(input)); } } [TestMethod] public void CheckException() { int[] input = new int[0]; Assert.ThrowsException<Exception>(() => MyTask.Calculate(input)); } ```
6	``` [TestMethod] public void CheckMinAbs() { ```

```
    int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 };
    Assert.AreEqual(1, MyTask.Calculate(input1));

    int[] input2 = new int[] { -7, 5, 3, 6, 2 };
    Assert.AreEqual(2, MyTask.Calculate(input2));

    int[] input3 = new int[] { 10, 25, -4, -32, 19 };
    Assert.AreEqual(4, MyTask.Calculate(input3));
}

[TestMethod]
public void CheckRandMin()
{
    Random r = new Random();
    for (int i = 0; i < 100; i++)
    {
        int v = r.Next(-5, 7);
        int p = r.Next(5);
        int[] input = new int[] { r.Next(7, 10),
            r.Next(7, 10), r.Next(7, 10),
            r.Next(7, 10), r.Next(7, 10) };
        input[p] = v;
        Assert.AreEqual(v, MyTask.Calculate(input));
    }
}

[TestMethod]
public void CheckMinusOne()
{
    int[] input = new int[0];
    Assert.AreEqual(-1, MyTask.Calculate(input));
}
```

7

```
[TestMethod]
public void CheckNumberPositive()
{
    int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 };
    Assert.AreEqual(6, MyTask.Calculate(input1));

    int[] input2 = new int[] { -7, 5, 3, 6, 2 };
    Assert.AreEqual(4, MyTask.Calculate(input2));

    int[] input3 = new int[] { 10, 25, -4, -32, 19 };
    Assert.AreEqual(3, MyTask.Calculate(input3));
}

[TestMethod]
public void CheckRandNumberPositive()
{
    Random r = new Random();
    for (int i = 0; i < 100; i++)
    {
       int v = r.Next(-5, 7);
       int n = r.Next(5);
       int s = r.Next(5);
       int[] input = new int[] { r.Next(-7, -1),
            r.Next(-7, -1), r.Next(-7, -1),
            r.Next(-7, -1), r.Next(-7, -1) };
       for (int k = 0; k < n; k++) input[(s + k) % input.Length] = r.Next(2, 10);
       Assert.AreEqual(n, MyTask.Calculate(input));
    }
}

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

	`int[] input = new int[0];` `Assert.AreEqual(0, MyTask.Calculate(input));` `}`
8	`[TestMethod]` `public void CheckNumberNegative()` `{` `int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 };` `Assert.AreEqual(0, MyTask.Calculate(input1));` `int[] input2 = new int[] { -7, 5, 3, 6, 2 };` `Assert.AreEqual(1, MyTask.Calculate(input2));` `int[] input3 = new int[] { 10, 25, -4, -32, 19 };` `Assert.AreEqual(2, MyTask.Calculate(input3));` `}` `[TestMethod]` `public void CheckRandNumberNegative()` `{` `Random r = new Random();` `for (int i = 0; i < 100; i++)` `{` `int v = r.Next(-5, 7);` `int n = r.Next(5);` `int s = r.Next(5);` `int[] input = new int[] { r.Next(-2, 10),` `r.Next(2, 10), r.Next(2, 10),` `r.Next(2, 10), r.Next(2, 10) };` `for (int k = 0; k < n; k++) input[(s + k) % input.Length] = r.Next(-7, -1);` `Assert.AreEqual(n, MyTask.Calculate(input));` `}` `}` `[TestMethod]` `public void CheckMinusOne()` `{` `int[] input = new int[0];` `Assert.AreEqual(-1, MyTask.Calculate(input));` `}`
9	`[TestMethod]` `public void CheckMax()` `{` `int[] input1 = new int[] { 2, 3, 8, 1, 2, 4};` `Assert.AreEqual(8, MyTask.Calculate(input1));` `int[] input2 = new int[] { -7, 5, 3, 6, 2 };` `Assert.AreEqual(6, MyTask.Calculate(input2));` `int[] input3 = new int[] { 10, 25, -4, -32, 19 };` `Assert.AreEqual(25, MyTask.Calculate(input3));` `}` `[TestMethod]` `public void CheckRandMax()` `{` `Random r = new Random();` `for (int i = 0; i < 100; i++)` `{` `int v = r.Next(7, 10);` `int p = r.Next(5);` `int[] input = new int[] { r.Next(-5, 7),` `r.Next(-5, 7), r.Next(-5, 7),` `r.Next(-5, 7), r.Next(-5, 7) };` `input[p] = v;` `Assert.AreEqual(v, MyTask.Calculate(input));`

	} } [TestMethod] public void CheckMinusOne() { int[] input = new int[0]; Assert.AreEqual(-1, MyTask.Calculate(input)); }
10	[TestMethod] public void CheckMaxIndex() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(2, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(3, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(1, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMaxIndex() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(7, 10); int p = r.Next(5); int[] input = new int[] { r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7) }; input[p] = v; Assert.AreEqual(p, MyTask.Calculate(input)); } } [TestMethod] public void CheckException() { int[] input = new int[0]; Assert.ThrowsException<Exception>(() => MyTask.Calculate(input)); }
11	[TestMethod] public void CheckAbsMax() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(8, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(7, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(32, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMax() { Random r = new Random(); for (int i = 0; i < 100; i++) {

```
        int v = r.Next(7, 10);
        int p = r.Next(5);
        int[] input = new int[] { r.Next(-5, 7),
            r.Next(-5, 7), r.Next(-5, 7),
            r.Next(-5, 7), r.Next(-5, 7) };
        input[p] = v;
        Assert.AreEqual(v, MyTask.Calculate(input));
    }
}

[TestMethod]
public void CheckZero()
{
    int[] input = new int[0];
    Assert.AreEqual(0, MyTask.Calculate(input));
}
```

12

```
[TestMethod]
public void CheckMin()
{
    int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 };
    Assert.AreEqual(1, MyTask.Calculate(input1));

    int[] input2 = new int[] { -7, 5, 3, 6, 2 };
    Assert.AreEqual(-7, MyTask.Calculate(input2));

    int[] input3 = new int[] { 10, 25, -4, -32, 19 };
    Assert.AreEqual(-32, MyTask.Calculate(input3));
}

[TestMethod]
public void CheckRandMin()
{
    Random r = new Random();
    for (int i = 0; i < 100; i++)
    {
        int v = r.Next(-5, 7);
        int p = r.Next(5);
        int[] input = new int[] { r.Next(7, 10),
            r.Next(7, 10), r.Next(7, 10),
            r.Next(7, 10), r.Next(7, 10) };
        input[p] = v;
        Assert.AreEqual(v, MyTask.Calculate(input));
    }
}

[TestMethod]
public void CheckMinusOne()
{
    int[] input = new int[0];
    Assert.AreEqual(-1, MyTask.Calculate(input));
}
```

13

```
[TestMethod]
public void CheckMinIndex()
{
    int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 };
    Assert.AreEqual(3, MyTask.Calculate(input1));

    int[] input2 = new int[] { -7, 5, 3, 6, 2 };
    Assert.AreEqual(0, MyTask.Calculate(input2));

    int[] input3 = new int[] { 10, 25, -4, -32, 19 };
    Assert.AreEqual(3, MyTask.Calculate(input3));
}

[TestMethod]
```

	public void CheckRandMinIndex() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(-5, 7); int p = r.Next(5); int[] input = new int[] { r.Next(7, 10), r.Next(7, 10), r.Next(7, 10), r.Next(7, 10), r.Next(7, 10) }; input[p] = v; Assert.AreEqual(p, MyTask.Calculate(input)); } } [TestMethod] public void CheckZero() { int[] input = new int[0]; Assert.AreEqual(0, MyTask.Calculate(input)); }
14	[TestMethod] public void CheckMinAbs() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(1, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(2, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(4, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMin() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(-5, 7); int p = r.Next(5); int[] input = new int[] { r.Next(7, 10), r.Next(7, 10), r.Next(7, 10), r.Next(7, 10), r.Next(7, 10) }; input[p] = v; Assert.AreEqual(v, MyTask.Calculate(input)); } } [TestMethod] public void CheckMinusOne() { int[] input = new int[0]; Assert.AreEqual(-1, MyTask.Calculate(input)); }
15	[TestMethod] public void CheckNumberPositive() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(6, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(4, MyTask.Calculate(input2));

```
    int[] input3 = new int[] { 10, 25, -4, -32, 19 };
    Assert.AreEqual(3, MyTask.Calculate(input3));
}

[TestMethod]
public void CheckRandNumberPositive()
{
    Random r = new Random();
    for (int i = 0; i < 100; i++)
    {
       int v = r.Next(-5, 7);
       int n = r.Next(5);
       int s = r.Next(5);
       int[] input = new int[] { r.Next(-7, -1),
            r.Next(-7, -1), r.Next(-7, -1),
            r.Next(-7, -1), r.Next(-7, -1) };
       for (int k = 0; k < n; k++) input[(s + k) % input.Length] = r.Next(2, 10);
       Assert.AreEqual(n, MyTask.Calculate(input));
    }
}

[TestMethod]
public void CheckException()
{
    int[] input = new int[0];
    Assert.ThrowsException<Exception>(() => MyTask.Calculate(input));
}
```

16

```
[TestMethod]
public void CheckNumberNegative()
{
    int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 };
    Assert.AreEqual(0, MyTask.Calculate(input1));

    int[] input2 = new int[] { -7, 5, 3, 6, 2 };
    Assert.AreEqual(1, MyTask.Calculate(input2));

    int[] input3 = new int[] { 10, 25, -4, -32, 19 };
    Assert.AreEqual(2, MyTask.Calculate(input3));
}

[TestMethod]
public void CheckRandNumberNegative()
{
    Random r = new Random();
    for (int i = 0; i < 100; i++)
    {
      int v = r.Next(-5, 7);
      int n = r.Next(5);
      int s = r.Next(5);
      int[] input = new int[] { r.Next(-2, 10),
            r.Next(2, 10), r.Next(2, 10),
            r.Next(2, 10), r.Next(2, 10) };
      for (int k = 0; k < n; k++) input[(s + k) % input.Length] = r.Next(-7, -1);
      Assert.AreEqual(n, MyTask.Calculate(input));
    }
}

[TestMethod]
public void CheckZero()
{
    int[] input = new int[0];
    Assert.AreEqual(0, MyTask.Calculate(input));
}
```

17

```
[TestMethod]
public void CheckMax()
{
    int[] input1 = new int[] { 2, 3, 8, 1, 2, 4};
    Assert.AreEqual(8, MyTask.Calculate(input1));

    int[] input2 = new int[] { -7, 5, 3, 6, 2 };
    Assert.AreEqual(6, MyTask.Calculate(input2));

    int[] input3 = new int[] { 10, 25, -4, -32, 19 };
    Assert.AreEqual(25, MyTask.Calculate(input3));
}

[TestMethod]
public void CheckRandMax()
{
    Random r = new Random();
    for (int i = 0; i < 100; i++)
    {
        int v = r.Next(7, 10);
        int p = r.Next(5);
        int[] input = new int[] { r.Next(-5, 7),
            r.Next(-5, 7), r.Next(-5, 7),
            r.Next(-5, 7), r.Next(-5, 7) };
        input[p] = v;
        Assert.AreEqual(v, MyTask.Calculate(input));
    }
}

[TestMethod]
public void CheckException()
{
    int[] input = new int[0];
    Assert.ThrowsException<Exception>(() => MyTask.Calculate(input));
}
```

18

```
[TestMethod]
public void CheckMaxIndex()
{
    int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 };
    Assert.AreEqual(2, MyTask.Calculate(input1));

    int[] input2 = new int[] { -7, 5, 3, 6, 2 };
    Assert.AreEqual(3, MyTask.Calculate(input2));

    int[] input3 = new int[] { 10, 25, -4, -32, 19 };
    Assert.AreEqual(1, MyTask.Calculate(input3));
}


[TestMethod]
public void CheckRandMaxIndex()
{
    Random r = new Random();
    for (int i = 0; i < 100; i++)
    {
        int v = r.Next(7, 10);
        int p = r.Next(5);
        int[] input = new int[] { r.Next(-5, 7),
        r.Next(-5, 7), r.Next(-5, 7),
        r.Next(-5, 7), r.Next(-5, 7) };
        input[p] = v;
        Assert.AreEqual(p, MyTask.Calculate(input));
    }
}
```

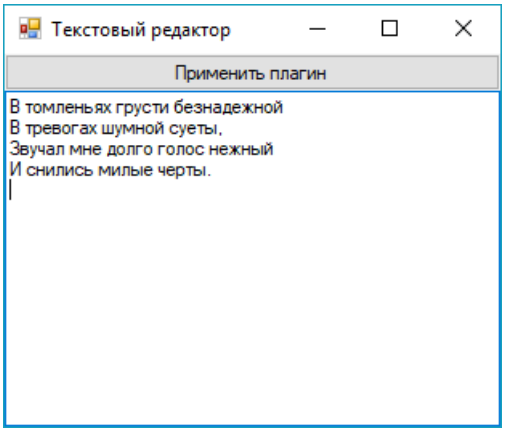
	[TestMethod] public void CheckZero() { int[] input = new int[0]; Assert.AreEqual(0, MyTask.Calculate(input)); }
19	[TestMethod] public void CheckAbsMax() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(8, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(7, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(32, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMax() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(7, 10); int p = r.Next(5); int[] input = new int[] { r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7) }; input[p] = v; Assert.AreEqual(v, MyTask.Calculate(input)); } } [TestMethod] public void CheckZero() { int[] input = new int[0]; Assert.AreEqual(0, MyTask.Calculate(input)); }
20	[TestMethod] public void CheckMin() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(1, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(-7, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(-32, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMin() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(-5, 7); int p = r.Next(5); int[] input = new int[] { r.Next(7, 10),

```
            r.Next(7, 10), r.Next(7, 10),
            r.Next(7, 10), r.Next(7, 10) };
        input[p] = v;
        Assert.AreEqual(v, MyTask.Calculate(input));
    }
}

[TestMethod]
public void CheckMinusOne()
{
    int[] input = new int[0];
    Assert.AreEqual(-1, MyTask.Calculate(input));
}
```

7.5. Разработать плагин к текстовому редактору в виде DLL-библиотеки

Необходимо создать DLL-библиотеку, которая будет использоваться в качестве плагина к текстовому редактору. Плагин должен осуществлять заданную операцию над входным текстом и возвращать результирующий текст.

Текстовый редактор уже написан, его код и сборка лежат по адресу https://github.com/Nordth/istu-priklad-practic-2020/tree/master/Day7/Task_TextEditor. В данном проекте изменять ничего не нужно. Вам нужно создать **новый** проект типа «*Библиотека классов (.NET Framework)*». В этом проекте вам необходимо в пространстве имен *Task_TextEditor* создать класс *Plugin* с методом *Execute*, который принимает на вход строку и возвращает изменённый текст (см. пример).



После сборки вашего проекта запустите текстовый редактор, введите текст и нажмите кнопку «Применить плагин». Откроется окно выбора файла, выберите в нем вашу скомпилированную библиотеку (DLL-файл). После выбора создастся экземпляр вашего класса *Plugin*, вызовется метод *Execute* и результат запишется в исходное текстовое поле.

Пример

Преобразование: вырезать все пробелы

Код плагина:

```
namespace Task_TextEditor
{
    public class Plugin
    {
        public string Execute(string input)
        {
            return input.Replace(" ", "");
        }
    }
}
```

Варианты

№	Преобразование	№	Преобразование
1	Перевести все буквы первого слова каждой строке в верхний регистр	11	Удалить все слова размером меньше, чем из 5 букв

2	Перевернуть каждое слово в строке (абг деж -> гба жед)	12	Удалить все слова, в которых меньше 5 согласных
3	Удалить знаки препинания	13	Перевести в верхний регистр все гласные
4	Поменять местами слова в каждой паре слов: аб вг де жз -> вг аб жз де	14	Перевести в верхний регистр все согласные
5	Удалить все согласные	15	Удалить все слова, в которых больше 5 согласных
6	Перевернуть строки (абг деж -> жед гба)	16	Перевести первую букву каждого слова в верхний регистр
7	Перемешать в случайном порядке все слова в строке	17	Перевести в верхний регистр каждую вторую букву слов: абв гдеж -> аБв гДеЖ
8	Удалить все слова без согласных букв	18	Удалить все слова размером больше, чем из 5 букв
9	Удалить все слова без гласных букв	19	Удалить первое и последнее слова в строках
10	Удаление каждого второго слова	20	Удалить все гласные