Лабораторная работа №1.

Введение в автономное тестирование

Цель работы

Приобретение практических навыков использования тестового каркаса NUnit для создания автономных тестов.

Задание на лабораторную работу

Создать и настроить в Visual Studio решение, состоящие из двух проектов: проект с модульными тестами, и проект с тестируемыми классами.

Изучить тестовый каркас NUnit, его атрибуты методы, и применить для создания: простых тестов проверки возвращаемых значений, параметризованных тестов, тестов для проверки исключений, тестов для проверки состояний. Реализовать в созданных проектах эти виды тестов и тестируемый код для них. Изучить инструменты Visual Studio для запуска тестов и анализа покрытия кода тестами.

В процессе выполнения работы уделить внимание соблюдению соглашения именования тестовых классов и тестовым методов, также организации структуры папок файлов исходного кода.

Порядок выполнения работы

1. Создать проект для автономных (модульных) тестов

1. Запустить Visual Studio
2. Выбрать создание проекта.
3. Выбрать тип проекта Библиотека классов .Net Standart или .Net Core.
4. Задать параметры проекты

   4.1 Указать имя проекта KTPO4310.Ivanov.UnitTest

   4.2 Указать расположение

   4.3 Указать имя решения KTPO4310.Ivanov

   4.4 Флаг Поместить решение и проект в одном каталоге не отмечать

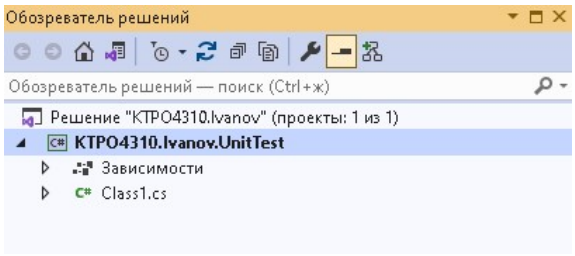
4.5 Нажать далее

Вместо 4310 – указать номер своей группы, вместо Ivanov свою фамилию и инициалы

5. Указать целевую платформу .Net Core 3.1

Нажать Создать

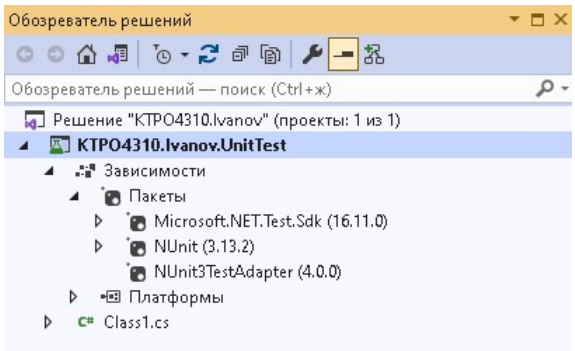
В результате получим решение



2. Подключить в проект «.UnitTest» тестовый каркас NUnit

1. Открыть Диспетчер пакетов Nuget
2. Источник пакета указать «nuget.org»
3. Переключиться на вкладку Обзор и набрать в строке Поиск “NUnit”.

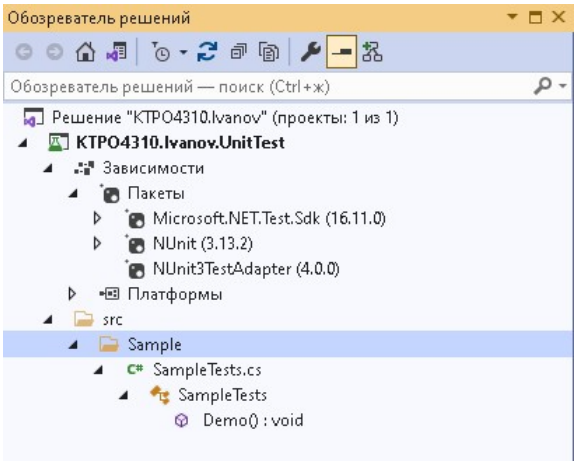
   Необходимо подключить пакеты NUnit и NUnit3TestAdapter
4. Установить пакет Microsoft.NET.Test.Sdk
5. В результате получим решение

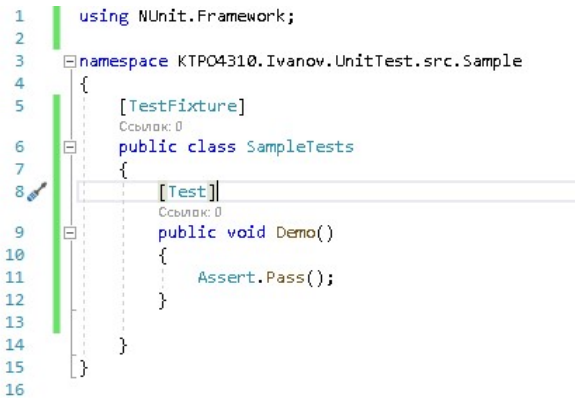


3. Добавить в проект первый модульный тест

1. Удалить файл Class1.cs
2. Добавить в проект папку src
3. Добавить в проект папку src\Sample
4. Добавить в папку src\Sample класс SampleTests
5. Сделать этот класс открытым (public)

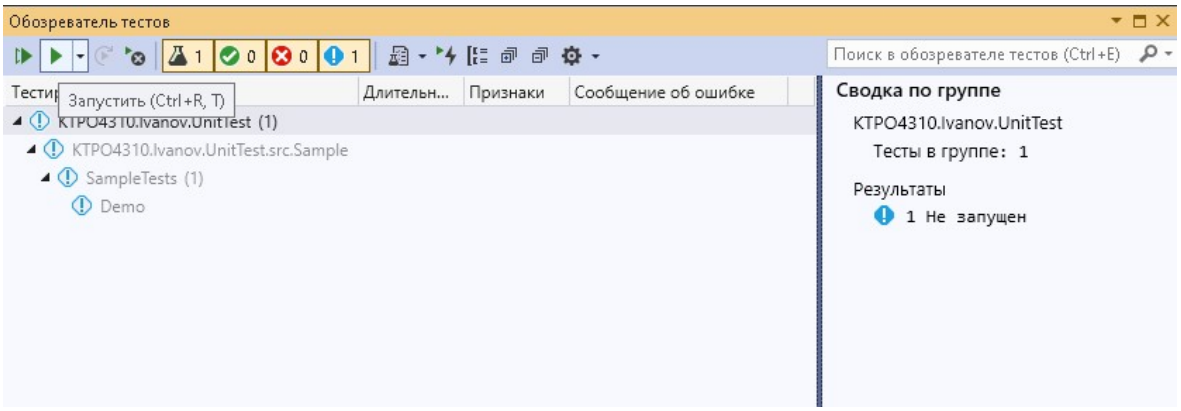
6. Добавить к классу атрибут [TestFixture] и ссылку на пространство имен NUnit.Framework;
7. Добавить в класс SampleTests метод открытый метод без параметров Demo.
   Добавить к метолу атрибут [Test]
   В тело метода добавить вызов Assert.Pass();
8. Удалить ненужные директивы uning
9. Перестроить решение. Убедиться, что проект компилируется без ошибок.
10. Результат показан на рисунках:



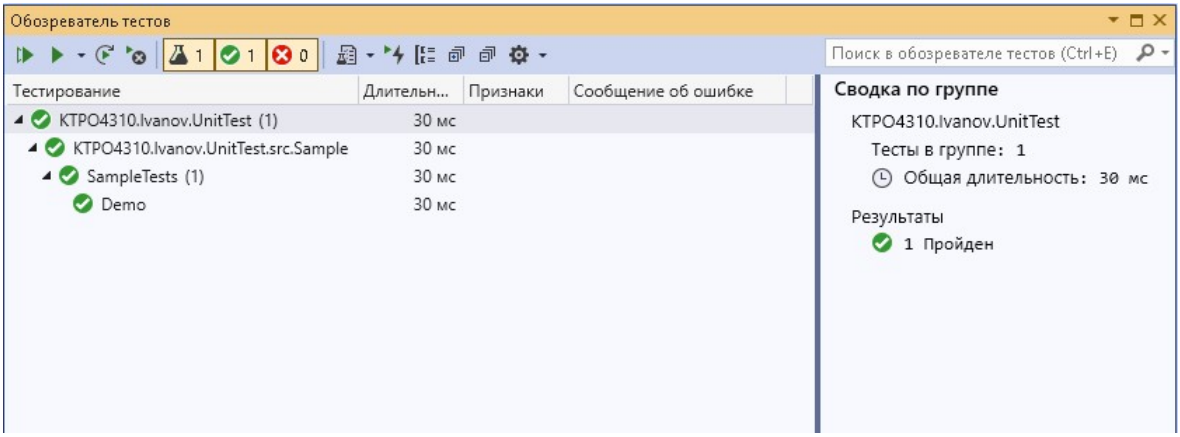


4. Запуск модульного теста

1. Открыть обозреватель тестов (Главное меню -> Test -> Обозреватель тестов).
2. Запустить тесты.

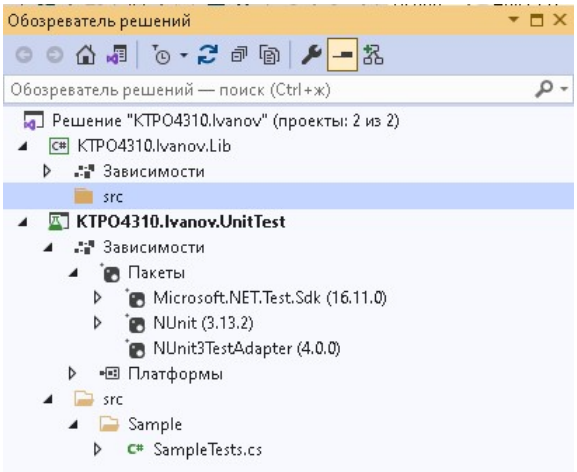


3. Если все предыдущие действия выполнены верно, получим результат, показанный на рисунке:



5. Добавить в решение проект для тестируемого кода

1. Добавить в решение новый проект Библиотека классов .Net Standart или .Net Core (Главное меню->Файл->Добавить->Создать проект).
   Имя проекта указать KTPO4310.Ivanov.Lib
   Целевую платформу указать .Net Core 3.1
2. Удалить файл Class1.cs
3. Добавить в проект папку src. Полученное решение в результате показано на рисунке:



6. Добавить тестируемый код и тесты для него

Добавить тестируемый код

1. Добавить в проект «.Lib» папку src\LogAn
2. Добавить в созданную класс LogAnalyzer
3. В класс добавить один публичный метод IsValidLogFileName, проверяющий правильность расширения файла. Вместо расширение файла “.SLF”, как на картинке используйте свою фамилию и инициалы заглавными буквами.

```
namespace KTPO4310.Ivanov.Lib.src.LogAn
{
    /// <summary>Анализато лог. файлов</summary>
    Ссылок: 0
    public class LogAnalyzer
    {
        /// <summary>Проверка правильности имени файла</summary>
        Ссылок: 0
        public bool IsValidLogFileName(string fileName)
        {
            if (fileName.EndsWith(".SLF"))
            {
                return false;
            }

            return true;
        }

    }
}
```

В методе присутствует ошибка, сохранить ее как показано на рисунке.

4. Перестроить решение.

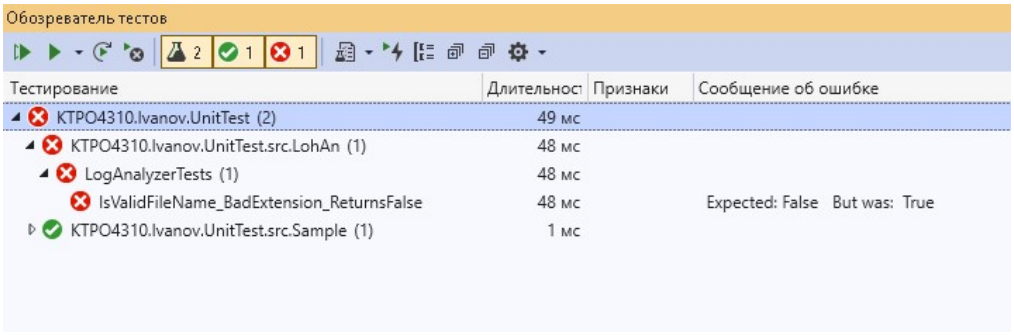
6. Создать тест для метода LogAnalyzer. IsValidLogFileName

1. Добавить в проект «.UnitTest» папку src\LogAn
2. Добавить в созданную папку класс LogAnalyzerTests
3. Пометить его атрибутом [TestFixture]
4. Добавьте в созданный класс метод тестирования IsValidLogFileName для сценария «Для неправильного расширении метод вернет false». Пометить его атрибутом [Test]
5. Добавить тестовый код как показано на рисунке

```
using KTPO4310.Ivanov.Lib.src.LogAn;
using NUnit.Framework;

namespace KTPO4310.Ivanov.UnitTest.src.LohAn
{
    [TestFixture]
    Ссылок: 0
    public class LogAnalyzerTests
    {
        [Test]
        | Ссылок: 0
        public void IsValidFileName_BadExtension_ReturnsFalse()
        {
            //Подготовка теста
            LogAnalyzer analyzer = new LogAnalyzer();

            //Воздействие на тестируемый объект
            bool result = analyzer.IsValidLogFileName("filewithbadextension.foo");

            //Проверка ожидаемого результата
            Assert.False(result);
        }
    }
}
```

6. Добавить ссылку на проект «.Lib»
7. Добавить в файл ссылку на пространство имен с классом LogAnalyzer
8. Построить решение.

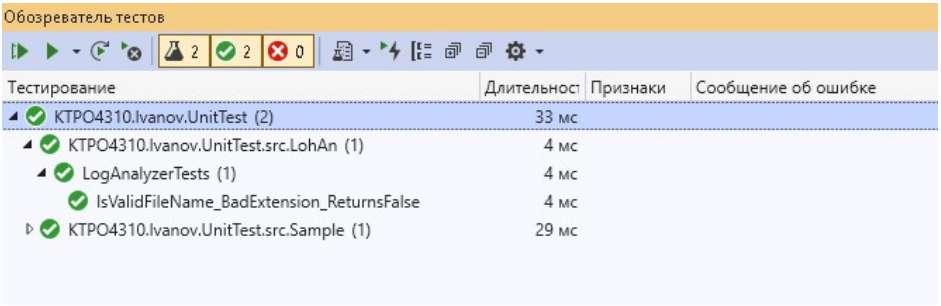
9. Запустить тесты.
10. Результат показан на рисунке.



Сделайте снимок окна и сохраните в документе MS Word

11. Исправить метод IsValidLogFileName, так чтобы тест выполнился.

Результат показан на рисунке.



Сделайте снимок окна и сохраните в документе MS Word

12. Добавьте самостоятельно положительные тесты для сценариев

«Для правильного расширения заглавными буквами метод вернет true»»

«Для правильного расширения строчными буквами метод вернет true»»

методы IsValidLogFileName_GoodExtensionUppercase_ReturnsTrue и IsValidLogFileName_GoodExtensionLowercase_ReturnsTrue соответственно.

Каков результат запуска теста? Сделайте снимок окна «Обозреватель тестов» и сохраните в документе MS Word

13. Исправить тестируемый метод, так чтобы тесты стали проходить.

Подсказка: если в метод EndsWith вторым параметром указать значение StringComparison.CurrentCultureIgnoreCase, то при сравнении строк величина букв учитываться не будет.

Каков результат запуска теста? Сделайте снимок окна «Обозреватель тестов» и сохраните в документе MS Word

7. Использование параметризованных тестов

В тестах IsValidLogFileName_GoodExtensionUppercase_ReturnsTrue и IsValidLogFileName_GoodExtensionLowercase_ReturnsTrue видно, что они отличаются только входными данными. Если вариантов входных параметров будет больше, то тесты станут трудоемкими для сопровождения.

Проведем рефакторинг тестов, используя параметризованные тесты.

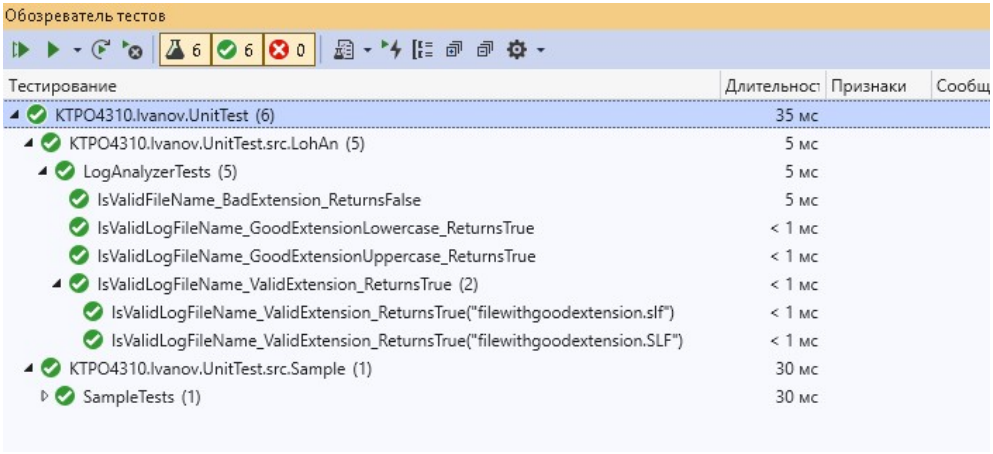
1. Скопируйте один из положительных тестов (например, IsValidLogFileName_GoodExtensionUppercase_ReturnsTrue) и дайте имя IsValidLogFileName_ValidExtension_ReturnsTrue (сценарий “Для правильного расширения метод вернет true”). Для параметризованного теста имя должно быть более общее.

   Примечание: в процессе рефакторинга обычно переименовывают и затем изменяют существующие тесты. Здесь в учебных целях исходные тесты сохраним.
2. Замените в новом тесте атрибут [Test] на [TestCase].
3. Превратите все входные значения тестов в параметры метода.
4. Поместите выявленные значения параметров в атрибуте.
5. Добавьте атрибут [TestCase] со значениями для каждого объединяемого теста.
6. На последнем шаге обычно удаляют тесты, которых на шаге 5 добавлен атрибут [TestCase]. Здесь в учебных целях исходные тесты сохраним.
7. Выполните тесты.

   Каков результат запуска тестов? Сделайте снимок окна «Обозреватель тестов» и сохраните в документе MS Word

Ожидаемый результат показан на рисунках:

```
[TestCase("filewithgoodextension.SLF")]
[TestCase("filewithgoodextension.slf")]
| Ссылок: 0
public void IsValidLogFileName_ValidExtension_ReturnsTrue(string file)
{
    LogAnalyzer analyzer = new LogAnalyzer();

    bool result = analyzer.IsValidLogFileName(file);

    Assert.True(result);
}
```



8. Проверка ожидаемых исключений. Анализ покрытия кода тестами

Путь тестируемый метод должен вызывать исключение, если в метод передано пустое имя файл.

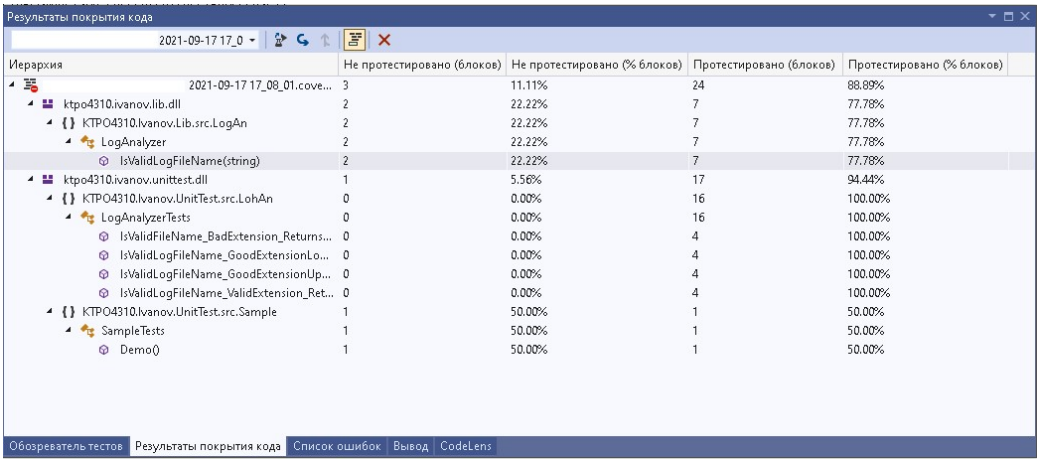
1. Добавьте в метод проверку:

```
if (string.IsNullOrEmpty(fileName))
{
    throw new ArgumentException("имя файла должно быть задано");
}
```

2. Скомпилируйте решение.
3. Выполните анализ покрытия кода тестами:

   Главное меню -> Test -> Анализ покрытия кода для всех тестов

   Каков результат запуска тестов? Сделайте снимок окна «Результаты покрытия кода» и сохраните в документе MS Word.



Щелкните два раза на методе IsValidLogFileName, чтобы просмотреть, какие фрагменты кода не покрыты тестами.

Сделайте снимок окна «Результаты покрытия кода» и сохраните в документе MS Word.

Ожидаемый результат показан на рисунке:

```
using System;

namespace KTPO4310.Ivanov.Lib.src.LogAn
{
    /// <summary>Анализато лог. файлов</summary>
    Ссылок: 8
    public class LogAnalyzer
    {
        /// <summary>Проверка правильности имени файла</summary>
        Ссылок: 4 | 5/5 пройдены
        public bool IsValidLogFileName(string fileName)
        {
            if (string.IsNullOrEmpty(fileName))
            {
                throw new ArgumentException("имя файла должно быть задано");
            }

            if (!fileName.EndsWith(".SLF",StringComparison.CurrentCultureIgnoreCase))
            {
                return false;
            }

            return true;
        }

    }
}
```

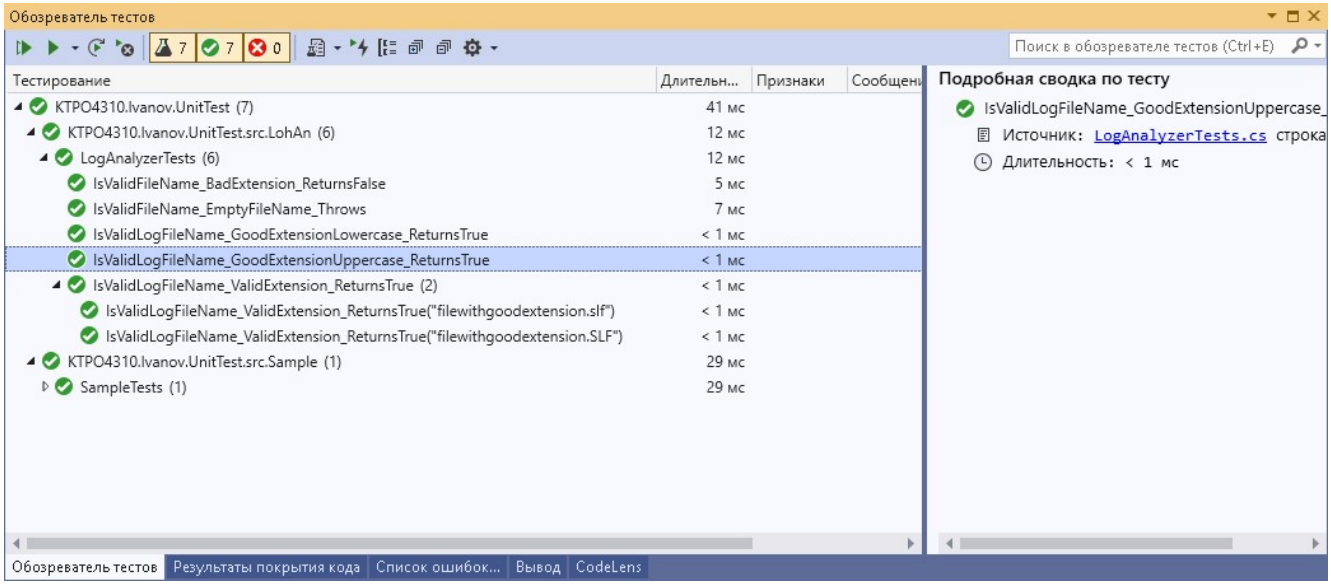
4. Создайте тестовый метод для сценария “Для пустого имения файла в методе вызывается исключение”, как показано на рисунке.

   Функция Assert.Catch перехватывает и возвращает исключение, которое было возбуждено внутри лямбда-выражения.

```
[Test]
| Ссылок: 0
public void IsValidFileName_EmptyFileName_Throws()
{
    //Подготовка теста
    LogAnalyzer analyzer = new LogAnalyzer();

    //Функция Catch перехватываает и возвращает исключение, которое было возбуждено внутри лямбда-выражения
    var ex = Assert.Catch<Exception>(() => analyzer.IsValidLogFileName(""));

    //Проверка, что исключение содержит ожидаемую строку
    StringAssert.Contains("имя файла должно быть задано", ex.Message);
}
```

5. Запустите тесты. Сделайте снимок окна «Результаты покрытия кода» и сохраните в документе MS Word.
6. Выполните анализ покрытия кода тестами. Сделайте снимок окна «Результаты покрытия кода» и сохраните в документе MS Word.

Ожидаемый результат показан рисунках.




```
using System;

namespace KTPO4310.Ivanov.Lib.src.LogAn
{
    /// <summary>Анализато лог. файлов</summary>
    Ссылок: 10
    public class LogAnalyzer
    {
        /// <summary>Проверка правильности имени файла</summary>
        Ссылок: 5 | 6/6 пройдены
        public bool IsValidLogFileName(string fileName)
        {
            if (string.IsNullOrEmpty(fileName))
            {
                throw new ArgumentException("имя файла должно быть задано");
            }

            if (!fileName.EndsWith(".SLF",StringComparison.CurrentCultureIgnoreCase))
            {
                return false;
            }

            return true;
        }

    }
}
```

9. Проверка изменения состояния

Пусть LogAnalyzer запоминает результат проверки имени файла.

1. Добавьте в тестируемый класс свойство WasLastFileNameValid

```
/// <summary>Анализато лог. файлов</summary>
Ссылок: 10
public class LogAnalyzer
{
    /// <summary>Результат предыдущей проверки имени файла</summary>
    Ссылок: 2
    public bool WasLastFileNameValid { get; set; }

    /// <summary>Проверка правильности имени файла</summary>
    Ссылок: 5 | 6/6 пройдены
    public bool IsValidLogFileName(string fileName)
    {
        WasLastFileNameValid = false;

        if (string.IsNullOrEmpty(fileName))
        {
            throw new ArgumentException("имя файла должно быть задано");
        }

        if (!fileName.EndsWith(".SLF",StringComparison.CurrentCultureIgnoreCase))
        {
            return false;
        }

        WasLastFileNameValid = true;

        return true;
    }

}
```

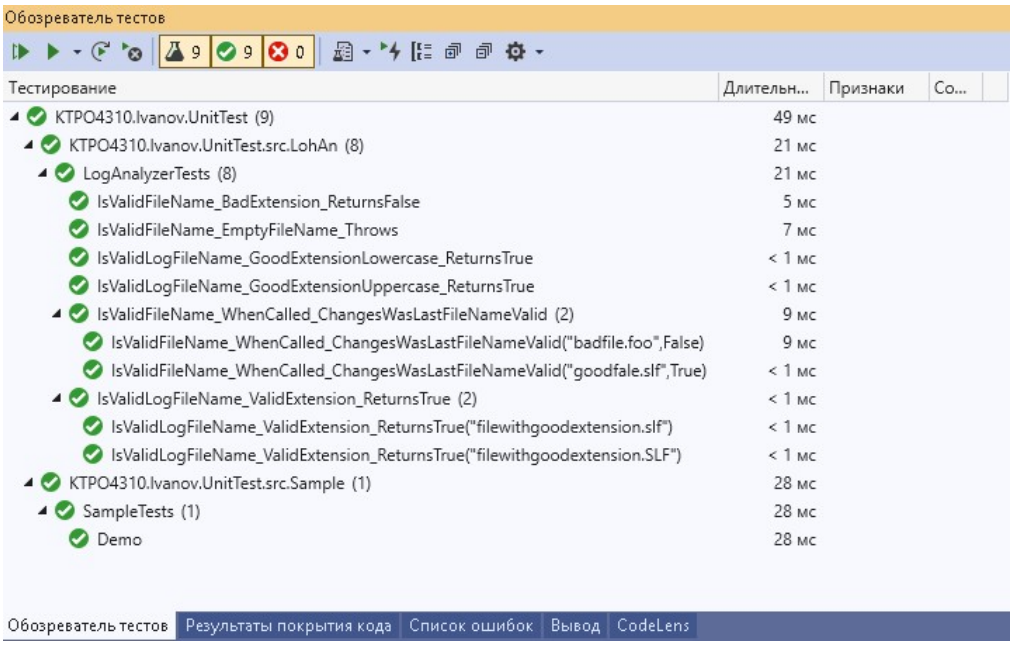
2. Создайте тестовый метод для сценария “При вызове метода меняется значение поля WasLastFileNameValid”. Используйте атрибуты [TestCase] для тестирования для правильных и неправильных имен.

```
[TestCase("badfile.foo", false)]
[TestCase("goodfale.slf", true)]
| Ссылок: 0
public void IsValidFileName_WhenCalled_ChangesWasLastFileNameValid(string file, bool expected)
{
    LogAnalyzer analyzer = new LogAnalyzer();

    analyzer.IsValidLogFileName(file);

    Assert.AreEqual(expected, analyzer.WasLastFileNameValid);
}
```

Обратите внимание, что в тесте не проверяется возвращаемый результат IsValidLogFileName, поскольку тест предназначен для другого аспекта поведения класса.

3. Запустите тесты. Сделайте снимок окна «Результаты покрытия кода» и сохраните в документе MS Word.

Ожидаемый результат показан на рисунке.



Содержание отчета

1. Постановка задачи.
2. Экранные формы с результатами выполнения заданий.
3. Выводы.