

Лабораторная работа №4.

Использование изолирующих каркасов

Цель работы

Приобретение практических навыков использования изолирующих каркасов для создания объектов заглушек и подставных объектов при автономном тестировании модулей, практика использования тестового каркаса NUnit, практика использования изолирующего каркаса NSubstitute.

Краткие теоретические сведения

Определение. Изолирующий каркас - это набор программируемых API, благодаря которым создавать поддельные объекты становится гораздо проще, быстрее и лаконичнее, чем вручную.

Определение. Динамический поддельный объект – это заглушка или подставка, создаваемая во время выполнения без необходимости кодировать реализацию вручную.

Задание на лабораторную работу

1. Подготовить учебный проект
2. Подключить изолирующий каркас
3. Изучить примы использования NSubstitute, реализовав демонстрационные тесты.
4. Создать реализацию тестов из лабораторной работы №2, заменив рукописные поддельные объекты на динамические.
5. Создать реализацию тестов из лабораторной работы №3, заменив рукописные поддельные объекты на динамические.
6. На каждом шаге делайте снимки исходного кода создаваемых или изменяемых классов и тестов, окна «Результаты тестов» и «Обозреватель решения» и сохраните в документе MS Word.

Порядок выполнения работы

1. Подготовка проекта

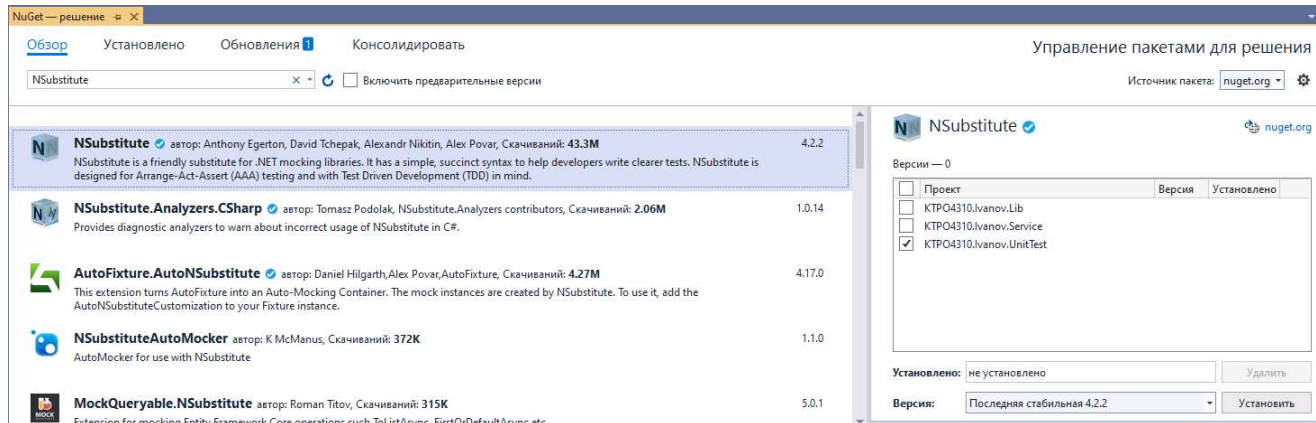
Для выполнения данной лабораторной работы возьмите решение, полученной в результате выполнения лабораторной работы №3.

Выполните тесты.

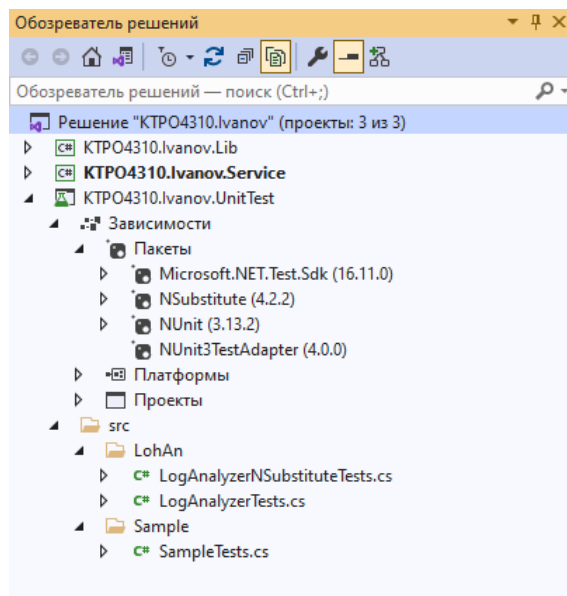
Зафиксируйте исходное состояние окна «Обозреватель решения», тестируемого класса и тестов, окна «Результаты тестов»: и сохраните в документе MS Word.

2. Подключить в проект «.UnitTest» изолирующий каркас NSubstitute

1. Открыть Диспетчер пакетов Nuget
2. Источник пакета указать «nuget.org»
3. Переключиться на вкладку Обзор и набрать в строке Поиск «NSubstitute».
4. Необходимо подключить пакет **NSubstitute**. Отметить его использование п проекте «.UnitTest»



5. В результате получим проект



3. Подделка значений, возвращаемых методом

1. Добавьте в папку Sample тестовый класс SampleNSubstituteTests.
2. Добавьте в класс SampleNSubstituteTests тестовый метод Returns_ParticularArg_Works(), который демонстрирует настройку возврата значения из поддельного объекта в зависимости от аргумента тестируемого метода. Тест показан на рисунке:

```
[Test]
public void Returns_ParticularArg_Works()
{
    //Создать поддельный объект
    IExtensionManager fakeExtensionManager = Substitute.For<IExtensionManager>();

    //Настроить объект, чтобы метод возвращал true для заданного значения входного параметра
    fakeExtensionManager.IsValid("validfile.ext").Returns(true);

    //Воздействие на тестируемый объект
    bool result = fakeExtensionManager.IsValid("validfil.ext");

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

3. Добавьте в класс SampleNSubstituteTests тестовый метод Returns_ArgAny_Works(), который демонстрирует настройку возврата значения из поддельного объекта, когда возвращаемый результат не зависит от аргумента тестируемого метода. Тест показан на рисунке:

```

[Test]
| Ссылка: 0
public void Returns_ArgAny_Works()
{
    //Создать поддельный объект
    IExtensionManager fakeExtensionManager = Substitute.For<IExtensionManager>();

    //Настроить объект, чтобы метод возвращал true независимо от параметров
    fakeExtensionManager.IsValid(Arg.Any<string>()).Returns(true);

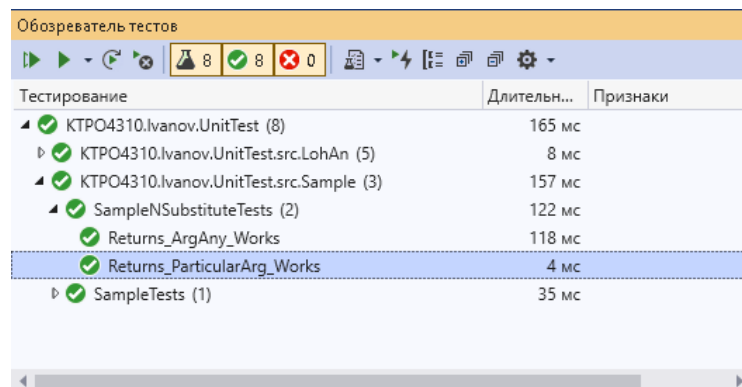
    //Воздействие на тестируемый объект
    bool result = fakeExtensionManager.IsValid("anyfile.ext");

    //Проверка ожидаемого результата
    Assert.IsTrue(result);
}

```

4. Выполните тесты.

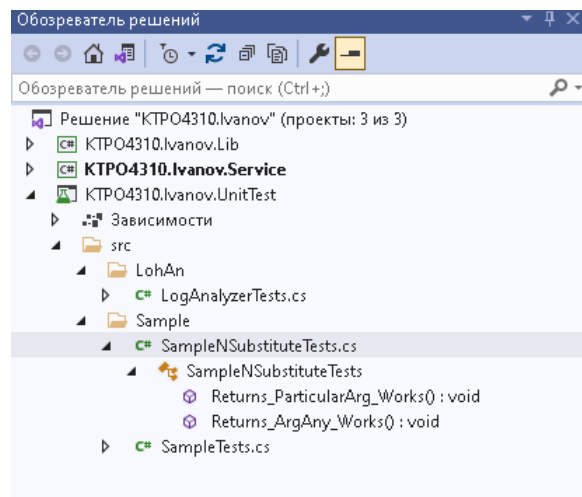
Ожидаемый результат.



Обозреватель тестов

Тестирование	Длительн...	Признаки
KTPO4310.Ivanov.UnitTest (8)	165 мс	
KTPO4310.Ivanov.UnitTest.src.LohAn (5)	8 мс	
KTPO4310.Ivanov.UnitTest.src.Sample (3)	157 мс	
SampleNSubstituteTests (2)	122 мс	
Returns_ArgAny_Works	118 мс	
Returns_ParticularArg_Works	4 мс	
SampleTests (1)	35 мс	

Ожидаемая структура проекта:



4. Имитация вызова исключений

- Добавьте в класс SampleNSubstituteTests тестовый метод Returns_ArgAny_Throws(), который демонстрирует настройку вызова исключения в поддельном объекте. Рассмотрен случай, когда результат не зависит от аргумента тестируемого метода. Тест показан на рисунке:

✔ | Ссылка: 0

Ожидаемый результат.

Ожидаемая структура проекта:

5. Проверка вызова поддельного объекта

```

[Test]
// Ссылка: 0
public void Received_ParticularArg_Saves()
{
    //Создать поддельный объект
    IWebService mockWebService = Substitute.For<IWebService>();

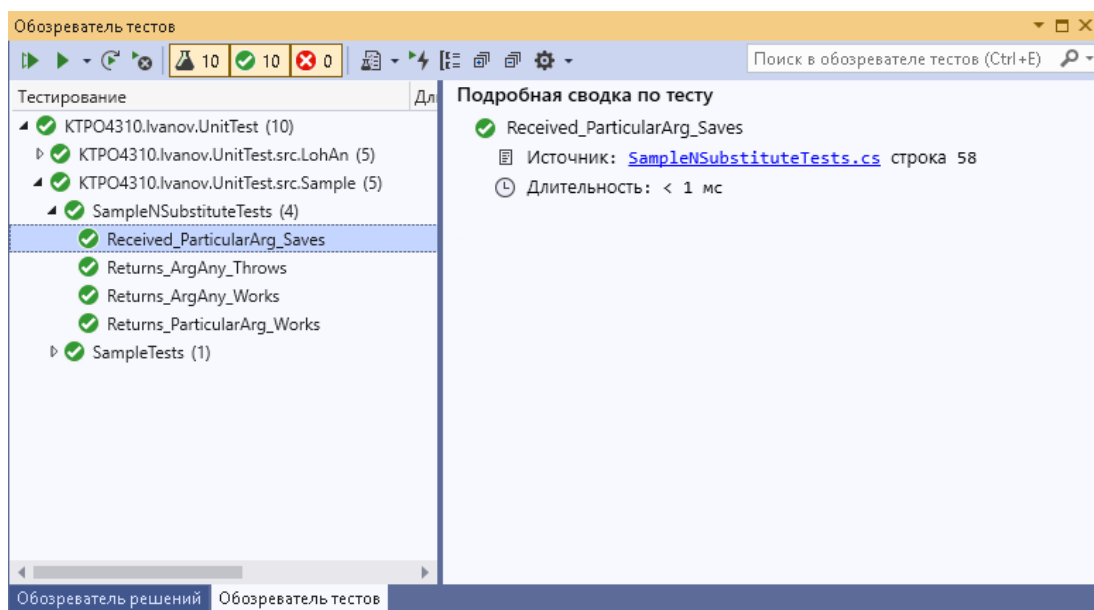
    //Воздействие на поддельный объект
    mockWebService.LogError("Поддельное сообщение");

    //Проверка, что поддельный объект сохранил параметры вызова
    mockWebService.Received().LogError("Поддельное сообщение");
}

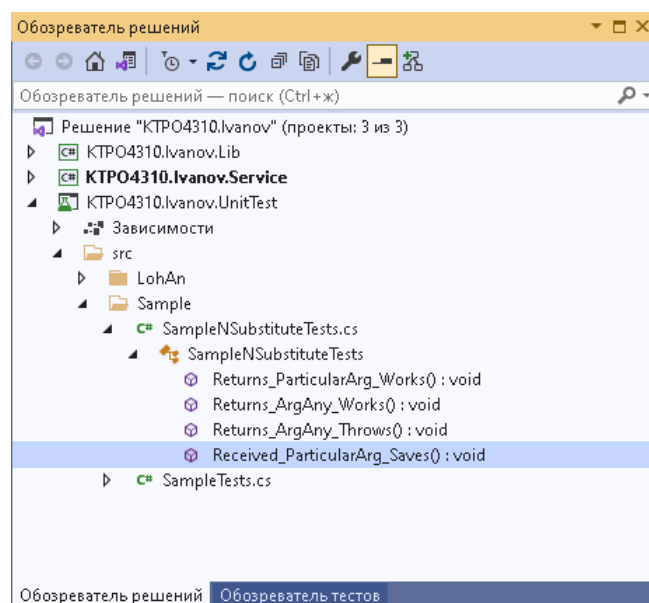
```

2. Выполните тесты

Ожидаемый результат.



Ожидаемая структура проекта:

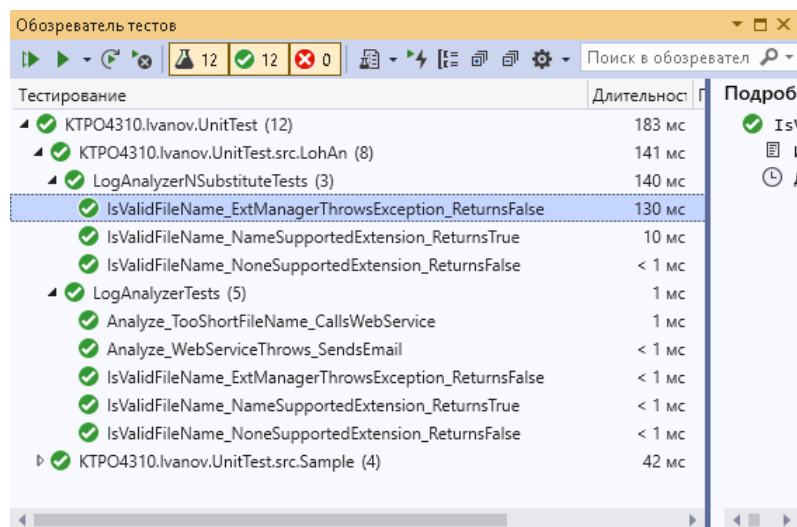
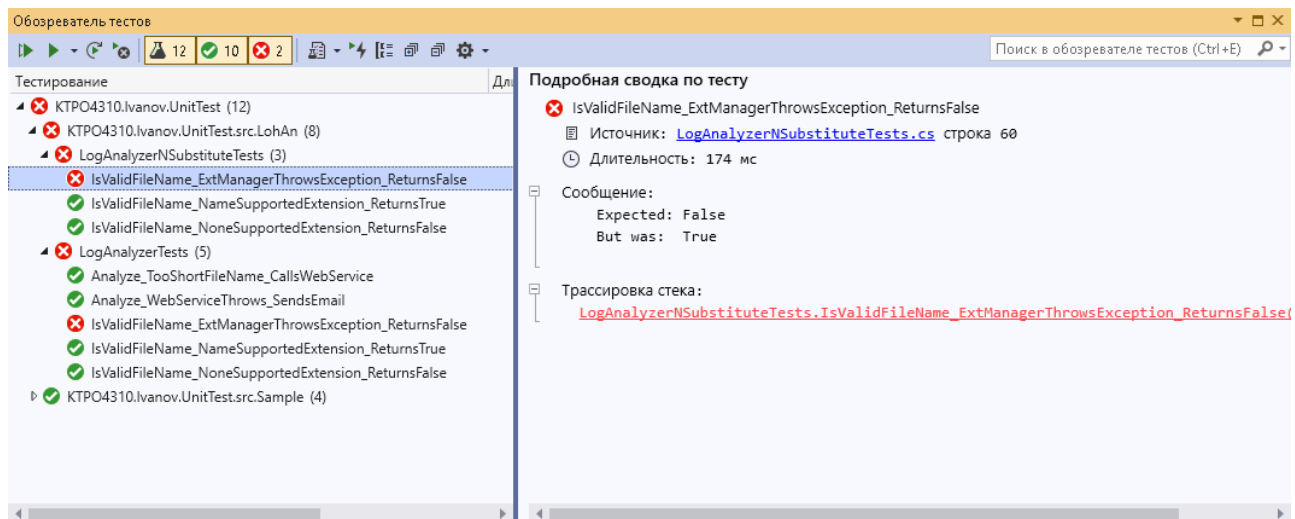


6. Замена рукописной подделки динамической

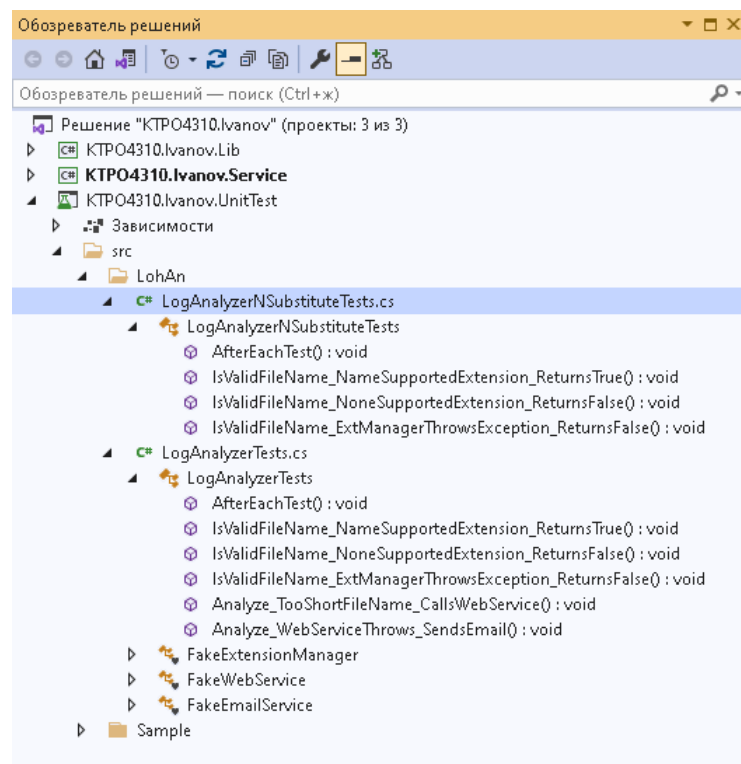
Во лабораторной работе №2 были созданы 3 теста использующие рукописные заглушки. Реализуйте эти тесты с использованием заглушек, созданных с помощью изолирующего каркаса. Используйте сведения, полученные в упражнениях 3 и 4.

1. Добавьте в папку LogAn новый тестовый класс LogAnalyzerNSubstituteTests. В нем разместим тесты для класса LogAnalyzer реализованных с использованием изолирующего каркаса.
2. Реализуйте тестовый метод в классе LogAnalyzerNSubstituteTests тестовый метод IsValidFileName_NameSupportedExtension_ReturnsTrue (см. лабораторную работу №2).
3. Реализуйте тестовый метод в классе LogAnalyzerNSubstituteTests тестовый метод IsValidFileName_NoneSupportedExtension_ReturnsFalse (см. лабораторную работу №2).
4. Реализуйте тестовый метод в классе LogAnalyzerNSubstituteTests тестовый метод IsValidFileName_ExtManagerThrowsException_ReturnsFalse (см. лабораторную работу №2).
5. Выполните тесты
6. Проверьте правильность последнего тестового метода. Для этого внесите в тестируемый метод дефект, ошибку которую должен обнаружить тест. Выполните тест и зафиксируйте результат, в том числе и текст сообщения об ошибке. Убедитесь, что вариант теста и рукописной и динамической заглушкой работают одинаково.
Восстановите правильный код.

Ожидаемый результат.



Ожидаемая структура проекта:



7. Совместное использование заглушки и подставки

Во лабораторной работе №3 были созданы 2 теста для метода Analyze класса LogAnalyzer, использующие рукописные заглушки и поддельные объекты. Реализуйте эти тесты с использованием поддельных объектов, созданных с помощью изолирующего каркаса. Используйте сведения, полученные в упражнениях 3 и 4.

1. Реализуйте тестовый метод в классе LogAnalyzerNSubstituteTests тестовый метод Analyze_TooShortFileName_CallsWebService() (см. лабораторную работу №3).
2. Реализуйте тестовый метод в классе LogAnalyzerNSubstituteTests тестовый метод Analyze_WebServiceThrows_SendsEmail() (см. лабораторную работу №3).
3. Выполните тесты
4. Проверьте правильность последнего тестового метода. Для этого внесите в тестируемый метод дефект, ошибку которую должен обнаружить тест: в тестируемом методе поменяйте местами значения параметров при вызове метода SendEmail. Выполните тест и зафиксируйте результат, в том числе и текст сообщения об ошибке. Убедитесь, что вариант теста и рукописными и динамическим поддельными объектами работают одинаково.

Восстановите правильный код.

Ожидаемый результат.

