**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.

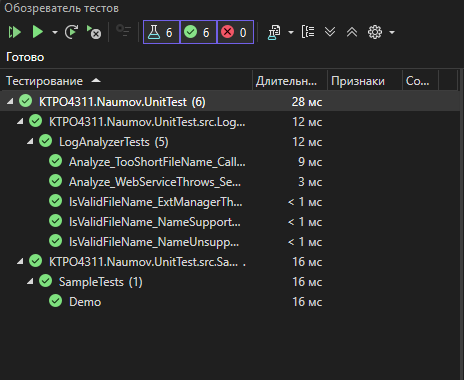
**Краткие теоретические сведения**

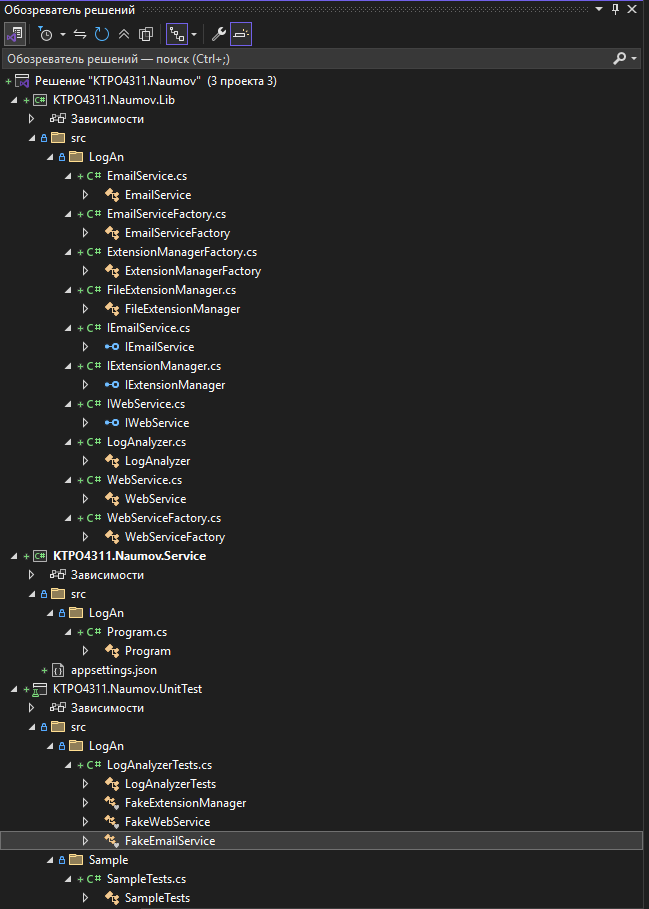
Определение. Изолирующий каркас - это набор программируемых API, благодаря которым создавать поддельные объекты становится гораздо проще, быстрее и лаконичнее, чем вручную.

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

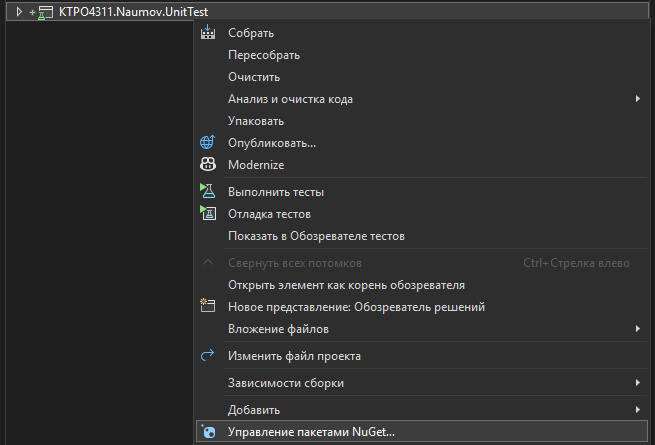
**Порядок выполнения работы**

1. **Подготовка проекта**
2. Берём решение, полученное в результате выполнения лабораторной работы № 3
3. Выполним тесты.
4. Фискируем исходное состояние проекта.





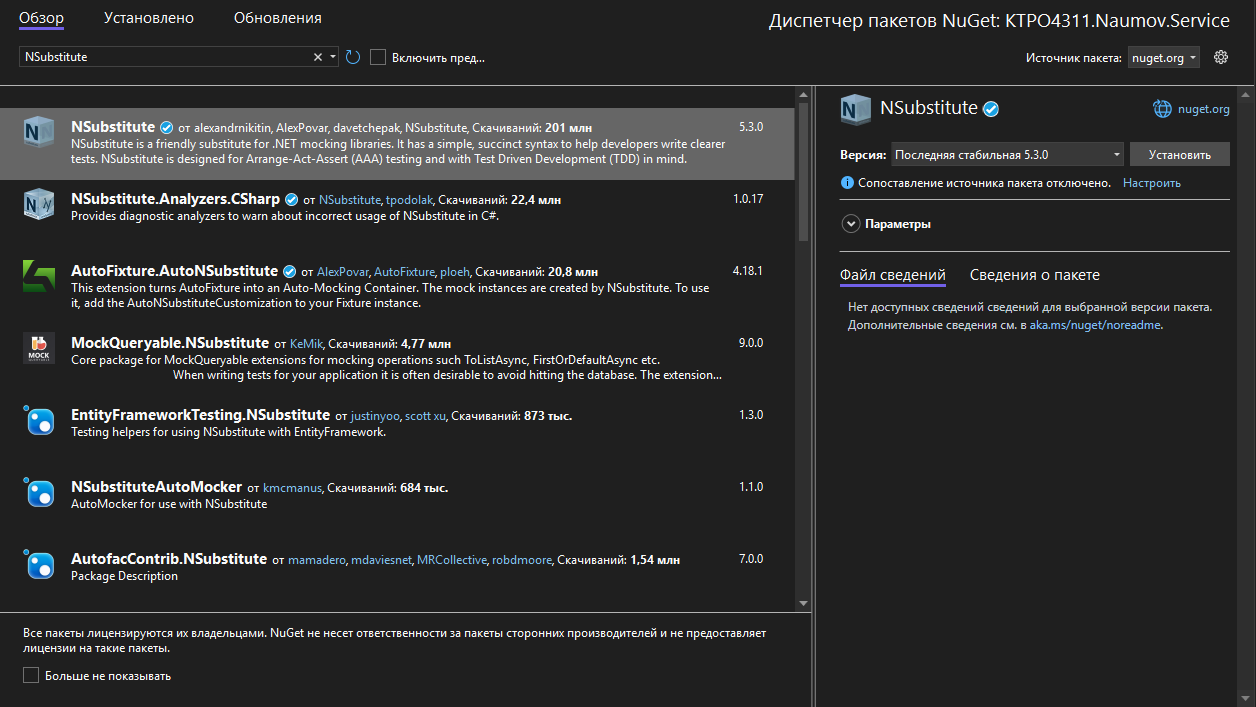
1. **Подключить в проект «.UnitTest» изолирующий каркас NSubstitute.**

****

**ПКМ ПК**

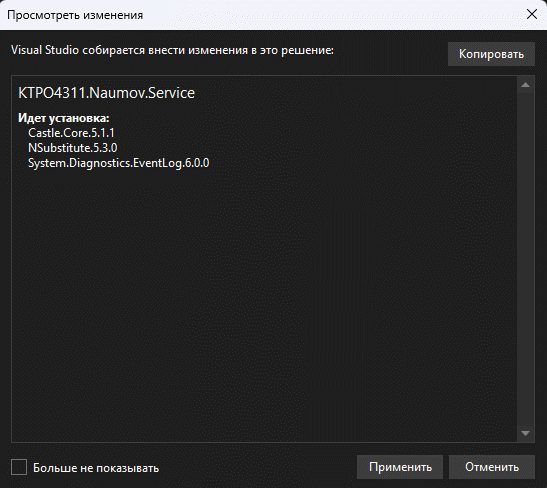
**2**

**1 ПК**

****

**3**

****

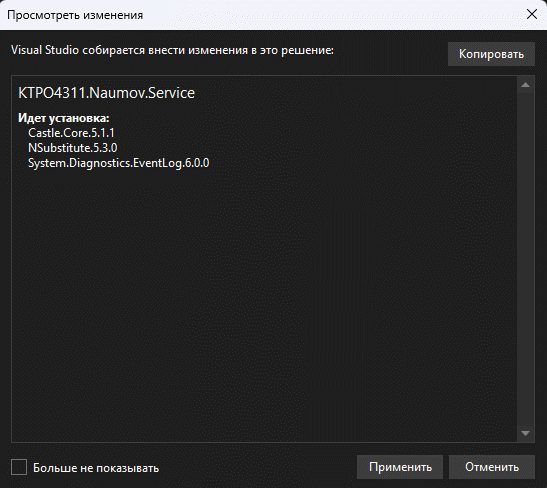
**5**

****

****

**4**

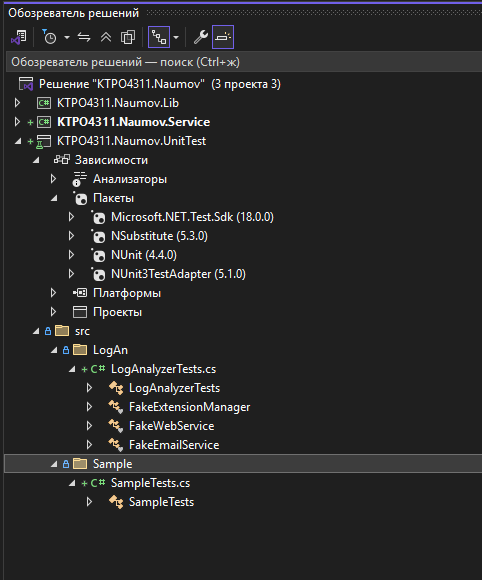
****

**6**

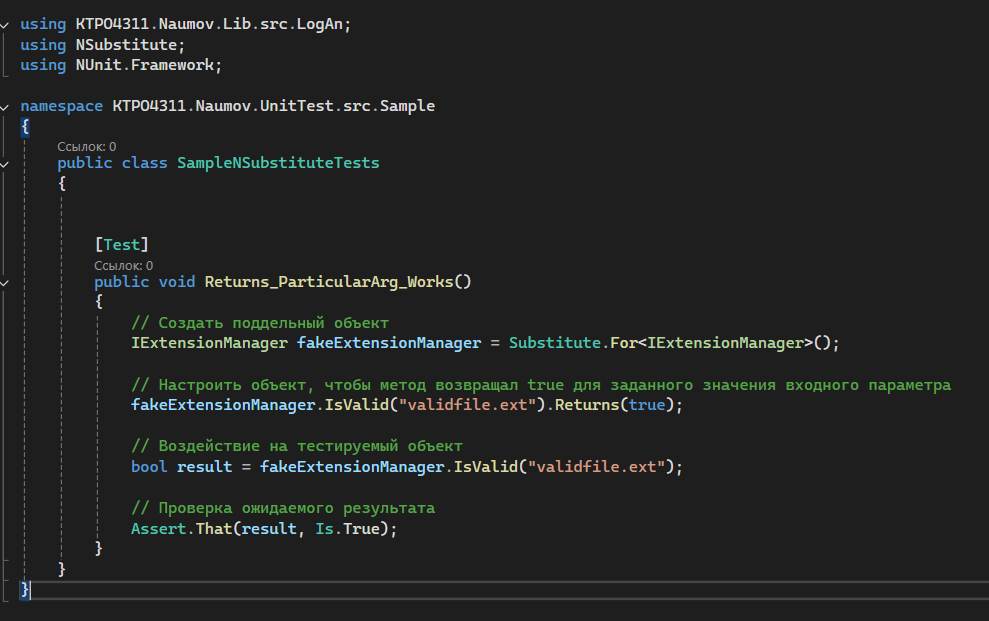
****

****

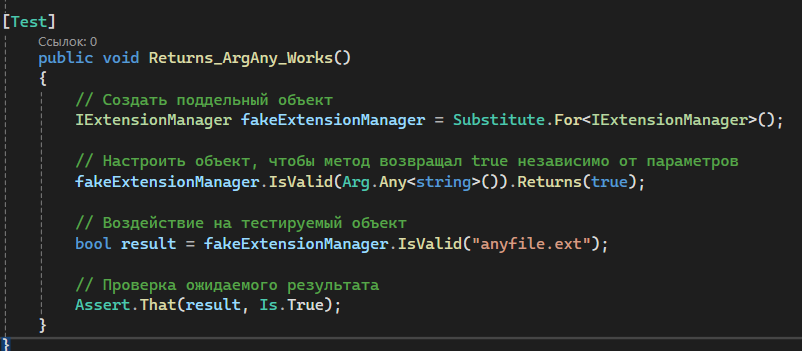
Получаем проект



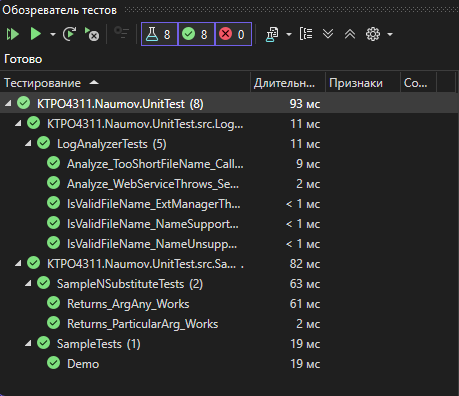
1. **Подделка значений, возвращаемых методом**
2. Создаём тестовый класс SampleNSubstituteTests
3. Создаём в новом классе метод Returns\_ParticularArg\_Works(). Метод проверяет, что поддельный объект IExtensionManager возвращает true при вызове IsValid("validfile.ext"), обеспечивая изоляцию кода и валидацию заданного поведения.

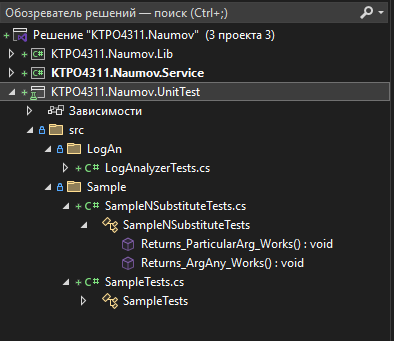


1. Создаём в новом классе метод Returns\_ArgAny\_Works(). Метод проверяет, что поддельный объект IExtensionManager возвращает true для IsValid независимо от аргумента, используя Arg.Any<string>(), и изолирует код для валидации поведения.

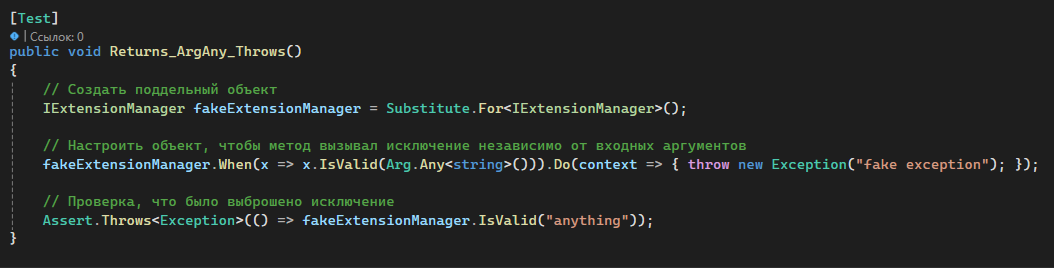


1. Выполним тесты.

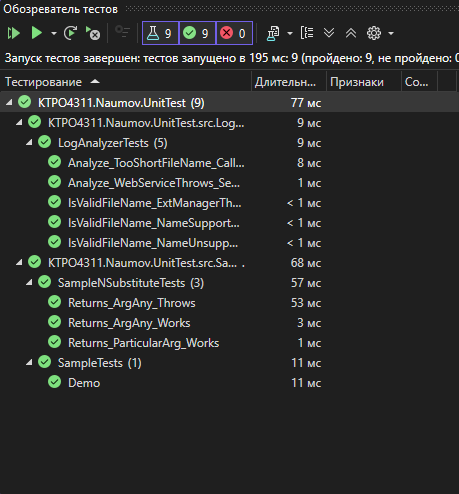


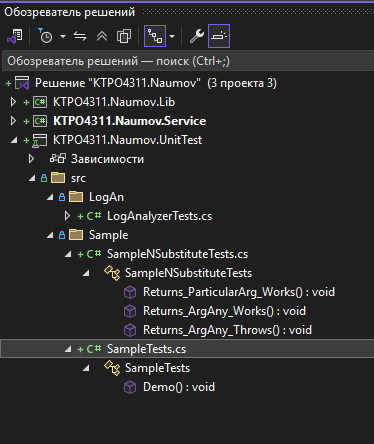


1. **Имитация вызова исключений.**
2. Добавим в класс метод Returns\_ArgAny\_Throws(), который проверяет, что поддельный объект IExtensionManager выбрасывает исключение Exception при вызове IsValid для любого аргумента, настроенного с помощью Arg.Any() и Do. Утверждение Assert.Throws подтверждает возникновение исключения.

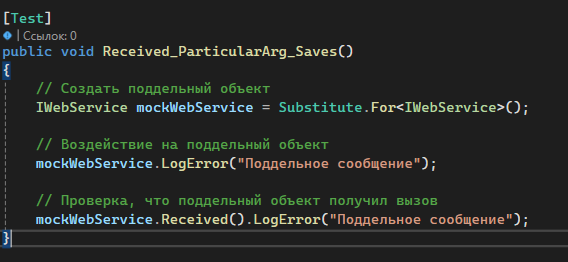


1. Выполним тесты

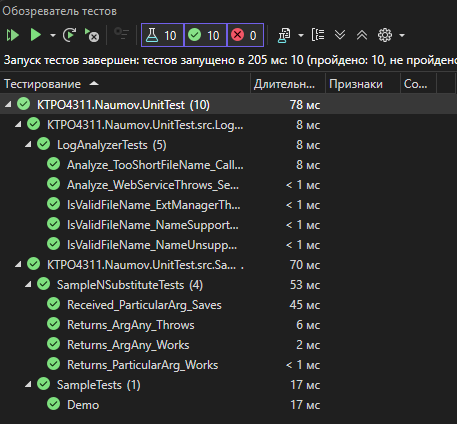


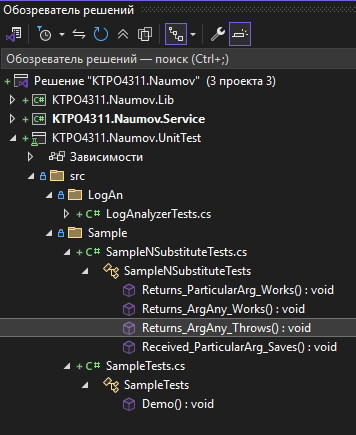


1. **Проверка вызова поддельного объекта.**
2. Добавим в класс метод Received\_ParticularArg\_Saves(), который проверяет, что поддельный объект IWebService, созданный с помощью NSubstitute, получил вызов метода LogError с конкретным аргументом "Поддельное сообщение". Утверждение Received() подтверждает выполнение вызова.

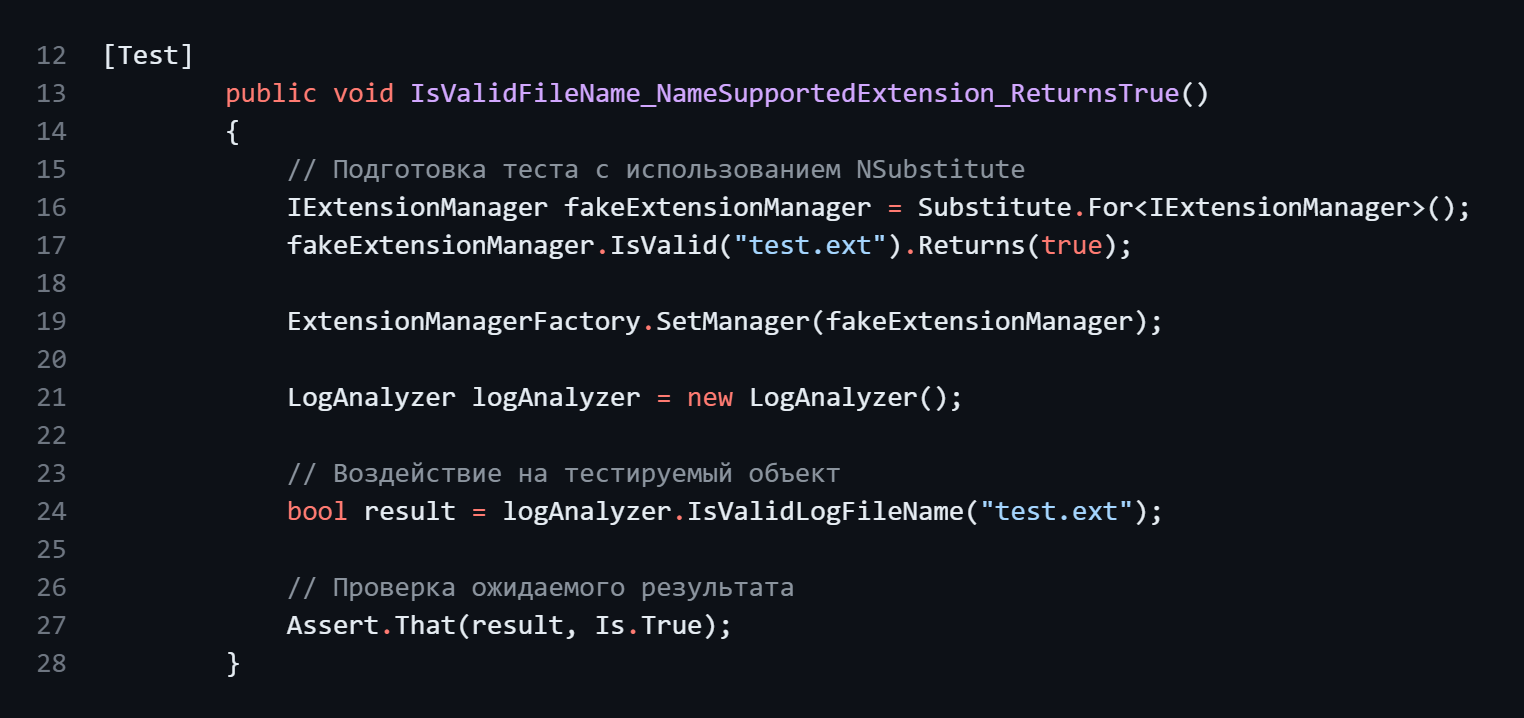


1. Выполним тесты.

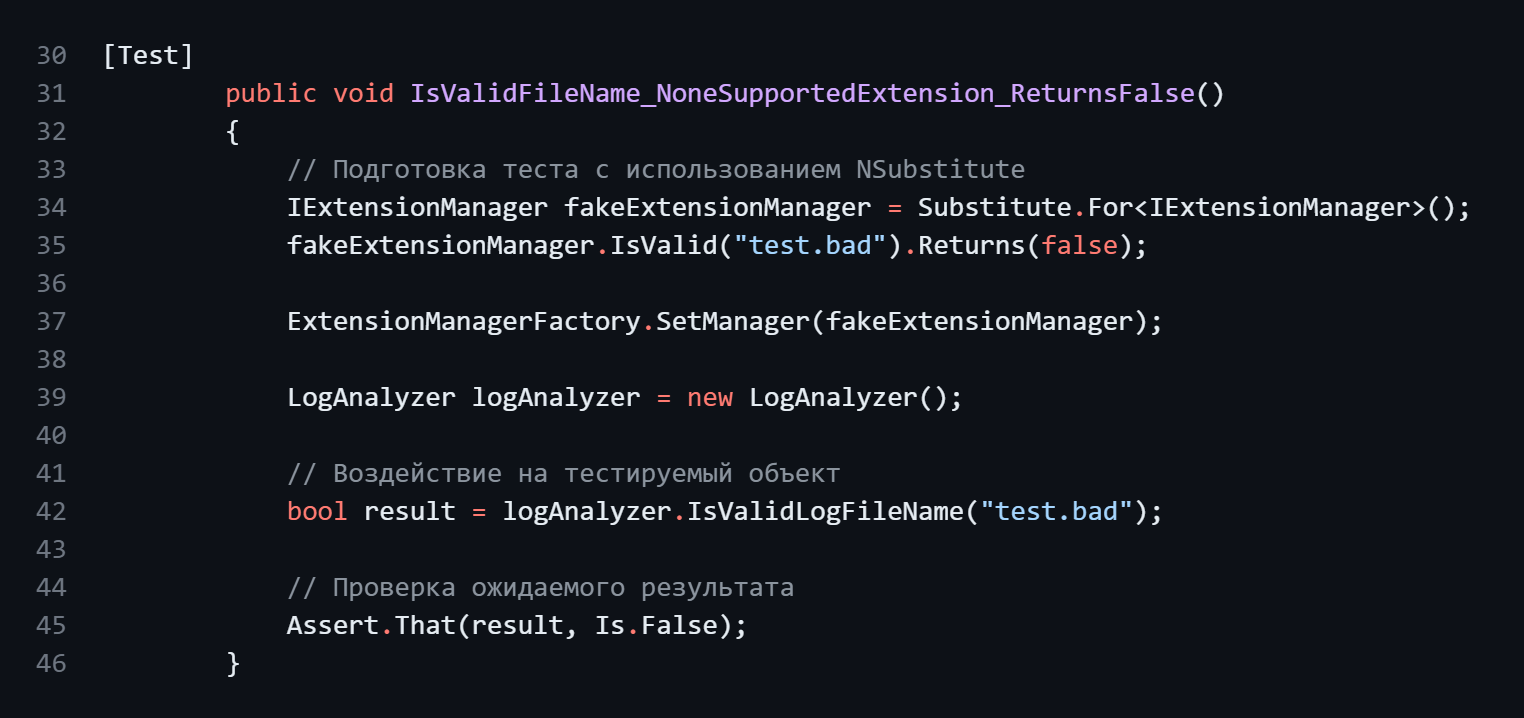




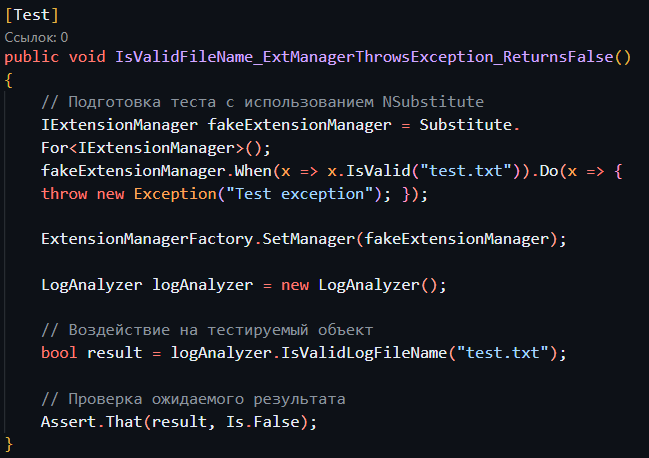
1. **Замена рукописной подделки динамической.**
2. Создаём в папке LogAn тестовый класс LogAnalyzerNSubstituteTests
3. В новом класс создаём тестовый метод IsValidFileName\_NameSupportedExtension\_ReturnsTrue()



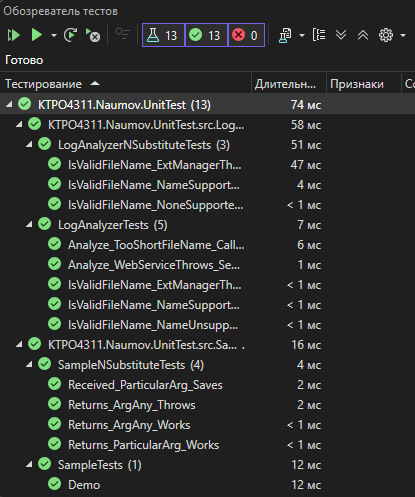
1. Создаём тестовый метод IsValidFileName\_NoneSupportedExtension\_ReturnsFalse()



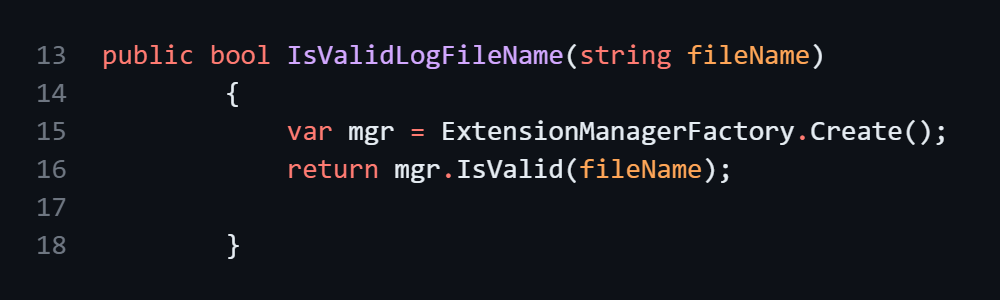
1. Создаём тестовый метод IsValidFileName\_ExtManagerThrowsException\_ReturnsFalse()



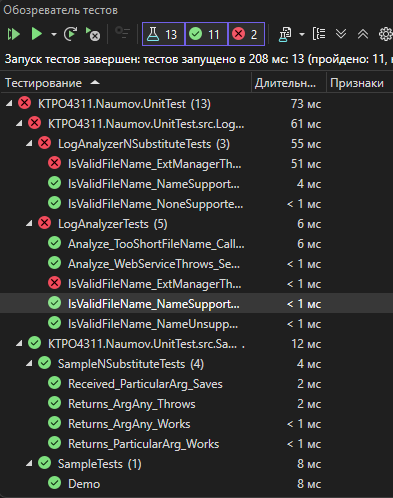
1. Выполним тесты

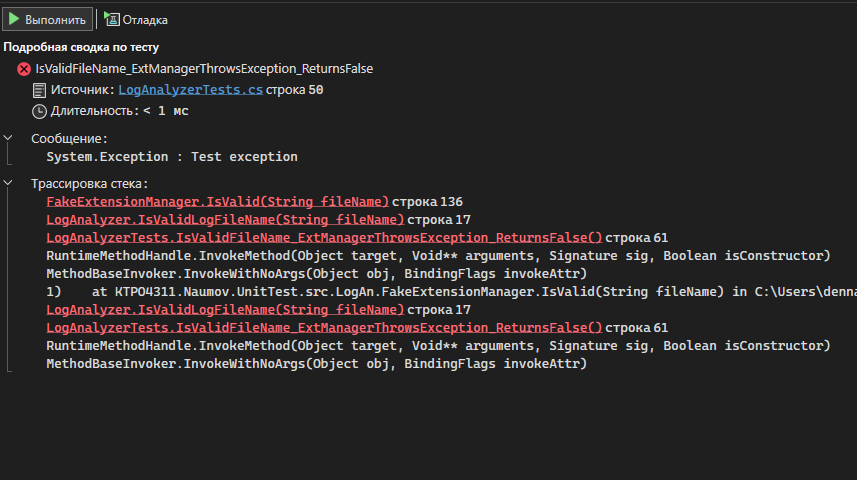


1. Проверим правильность последнего тестового метода. Внесём в тестируемый метод ошибку (дефект). Удалим блок try-catch.

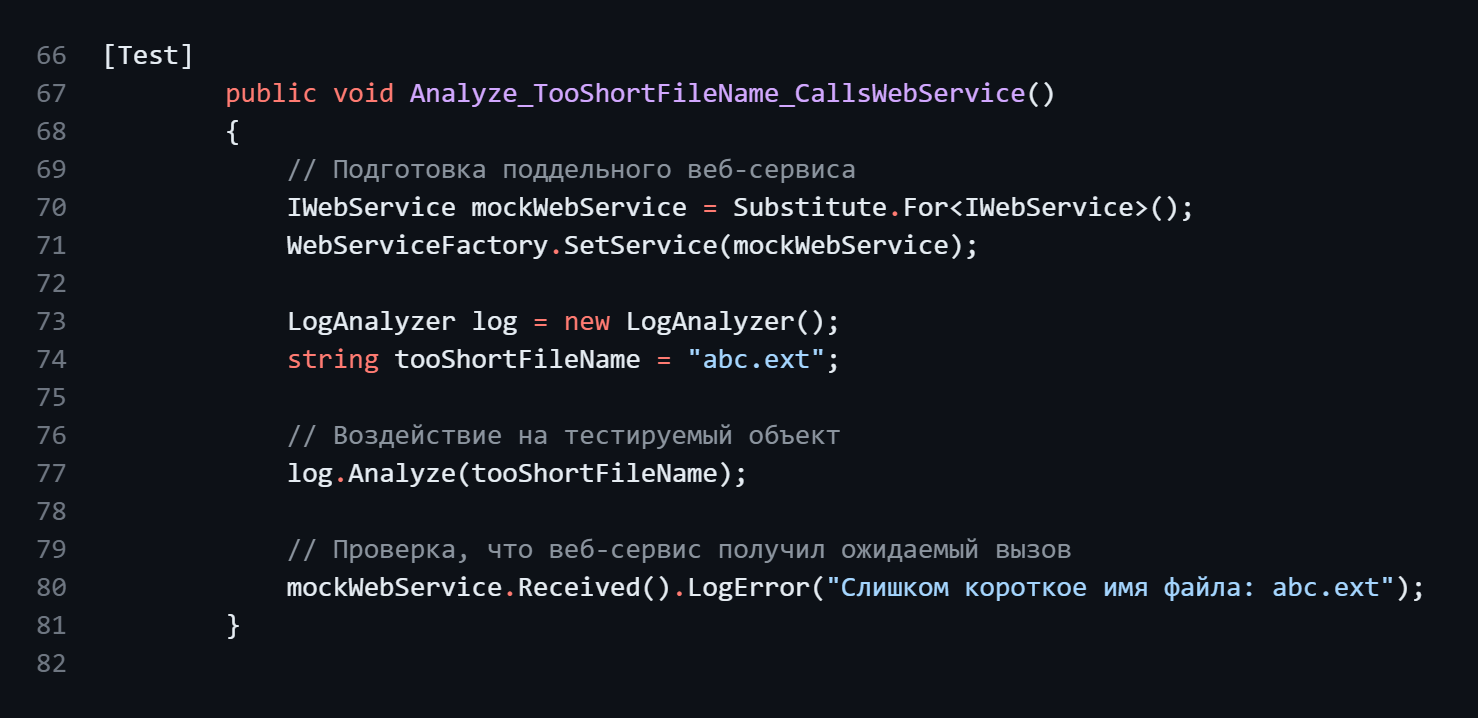


Выполним тесты

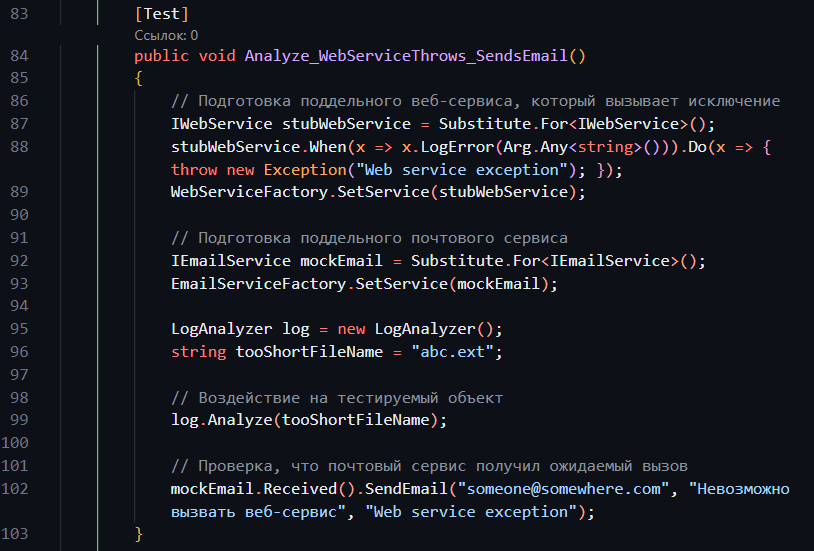




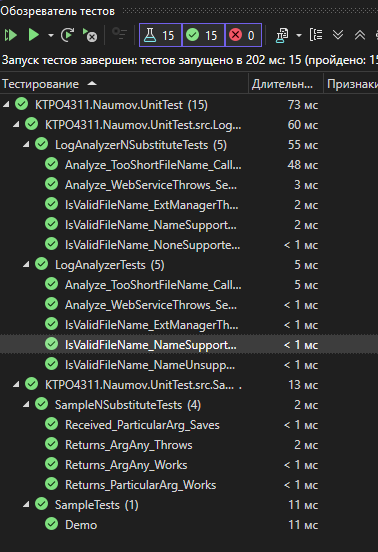
1. **Совместное использование заглушки и подставки.**
2. В классе LogAnalyzerNSubstituteTests реализуем тестовый метод Analyze\_TooShortFileName\_CallsWebService()



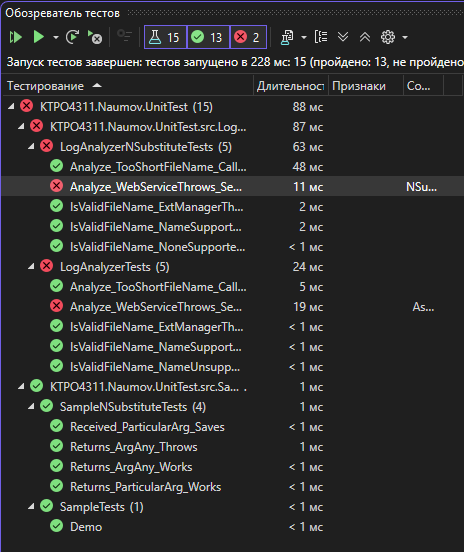
1. Здесь же создадим ещё один тестовый метод Analyze\_WebServiceThrows\_SendsEmail()

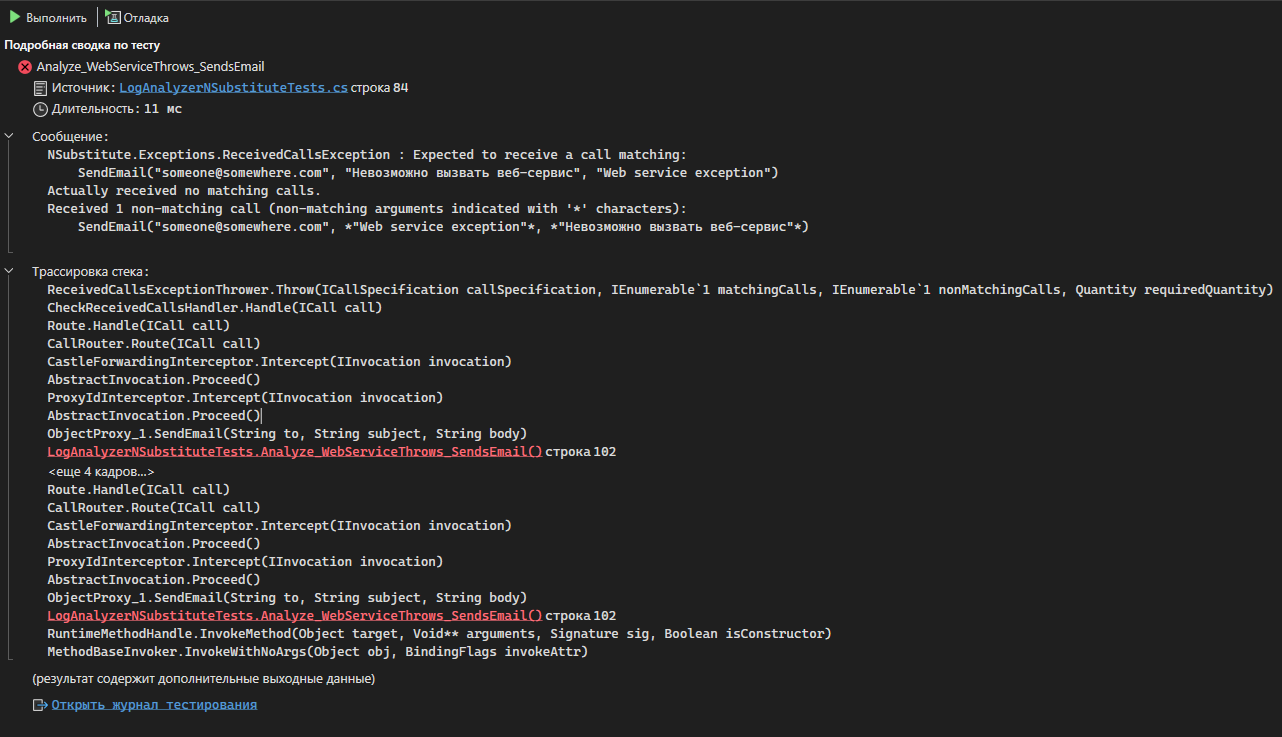


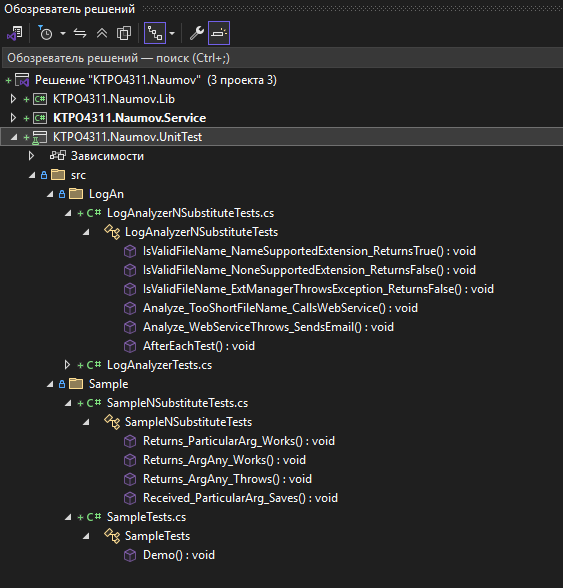
1. Выполним тесты



1. Проверим правильность последнего тестового метода. Внесём в тестируемый метод ошибку, которую должен обнаружить тест. В тестируемом методе поменяем местами значение параметров при вызове метода SendEmail.







**Выводы**

В ходе лабораторной работы №4 были приобретены практические навыки использования изолирующих каркасов для создания объектов заглушек и подставных объектов при автономном тестировании модулей. Цель работы заключалась в изучении и применении тестового каркаса NUnit и изолирующего каркаса NSubstitute для создания динамических поддельных объектов.

Изолирующий каркас представляет собой набор программируемых API, благодаря которым создавать поддельные объекты становится гораздо проще, быстрее и лаконичнее, чем вручную. Динамический поддельный объект - это заглушка или подставка, создаваемая во время выполнения без необходимости кодировать реализацию вручную.

В результате работы был создан тестовый класс LogAnalyzerNSubstituteTests с использованием изолирующего каркаса NSubstitute, реализованы пять тестовых методов для проверки различных сценариев работы класса LogAnalyzer, включая обработку поддерживаемых и неподдерживаемых расширений файлов, обработку исключений, а также тестирование взаимодействия с веб-сервисом и электронной почтой. Все тесты успешно выполняются, что подтверждает правильность использования изолирующих каркасов для автономного тестирования модулей.