**Лабораторная работа №5**

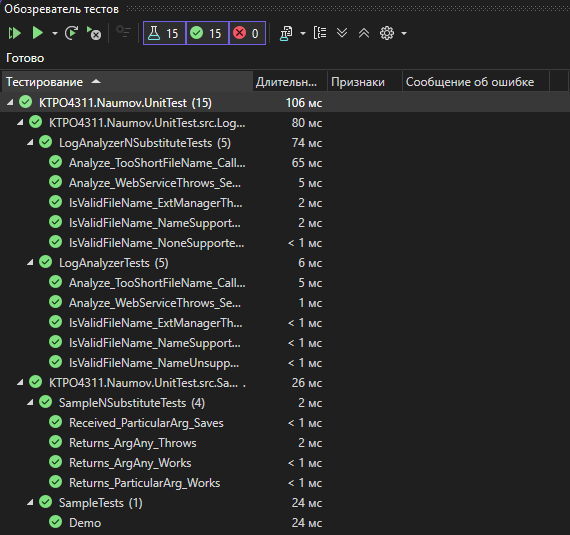
**Тестирование событий**

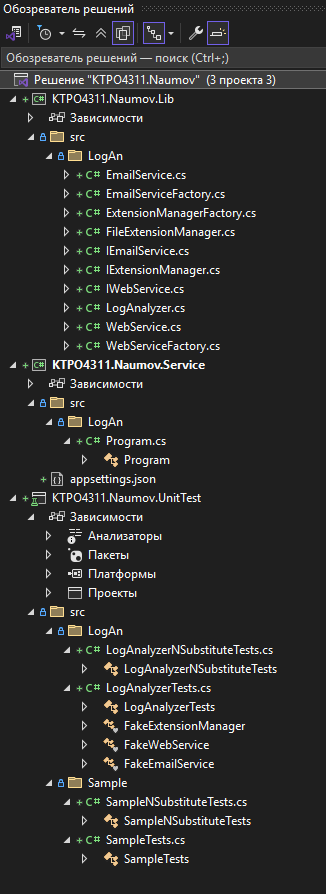
**Цель работы:**

Приобретение практических навыков использования делегатов (delegate) и событий (event); тестирования операций, связанных с событием; практика использования тестового каркаса NUnit, практика использования изолирующего каркаса NSubstitute.

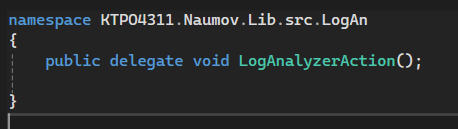
**Порядок выполнения работы**

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

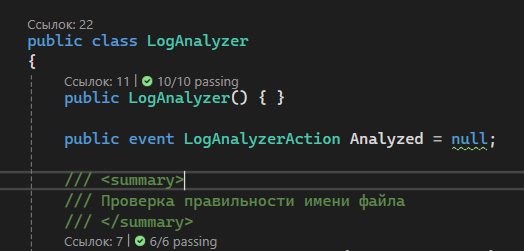




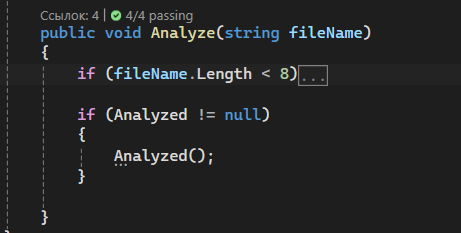
1. **Генерация события**
2. В проекте .Lib объявим делегат



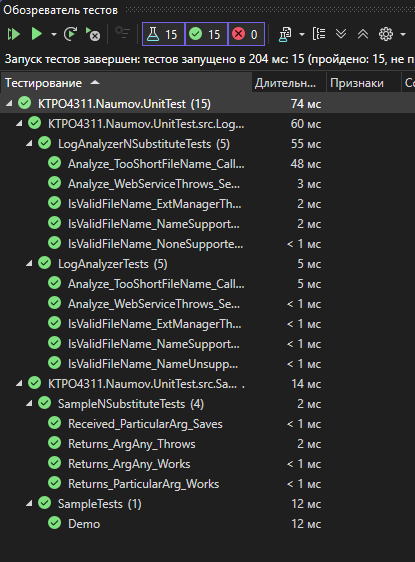
1. Объявляем событие в LogAnalyzer



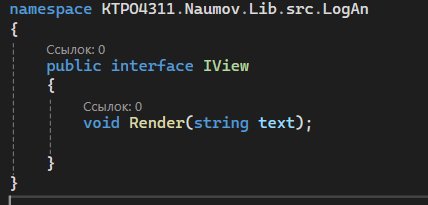
1. Добавляем вызов события в метод Analyze



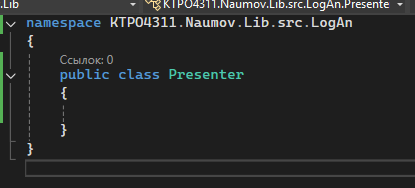
1. Выполним тесты.



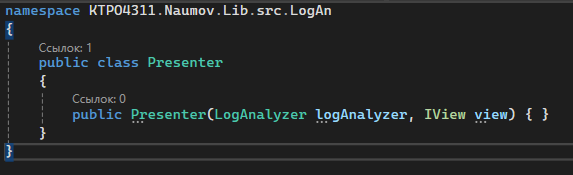
1. **Подписка объекта на события**
2. Добавим в проект .Lib интерфейс IVIew для представления. Создадим в этом интерфейсе метод Render.



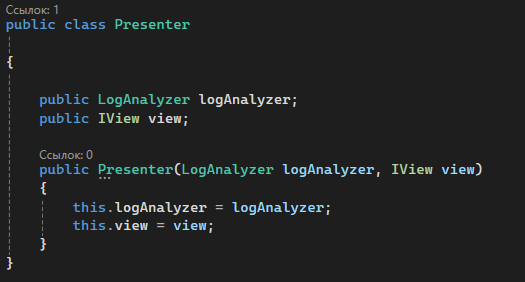
1. Создадим в этом же проекте класс Presenter.



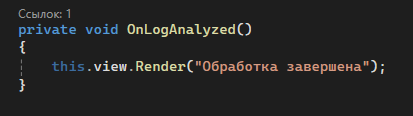
1. Реализуем в нём конструктор с 2-мя параметрами.



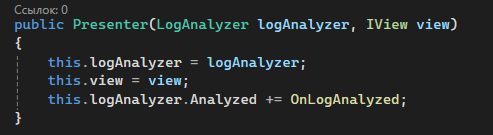
1. Добавим два поля в класс и инициализируем их в конструкторе.



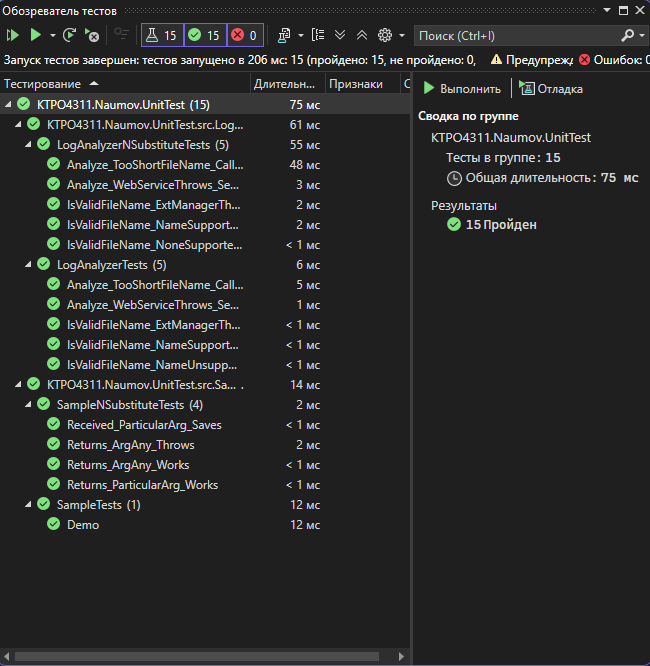
1. Создаём приватный метод OnLogAnalyzed(). Добавляем в него вызов обращение к представлению.

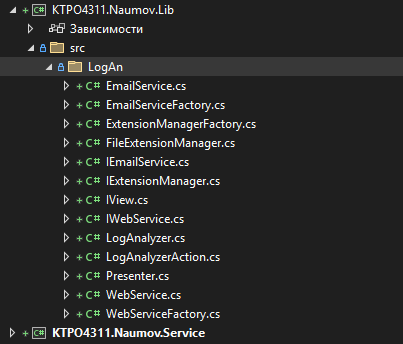


1. В конструкторе добавляем подписку на событие.

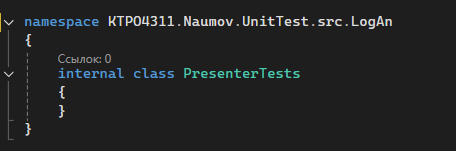


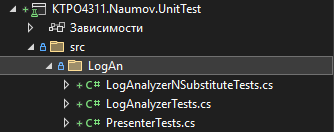
1. Выполним тесты



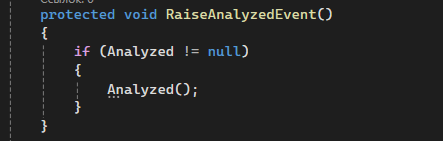


1. **Тестирование и прослушивание события. Введение зазора с помощью наследования.**
2. Создаём класс PresenterTests

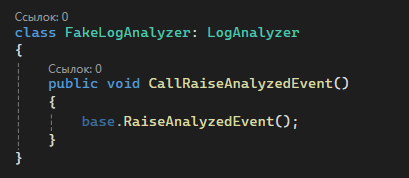




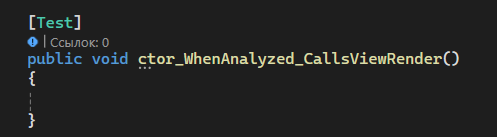
1. Создаём метод RaiseAnalyzedEvent() в LogAnalyzer.



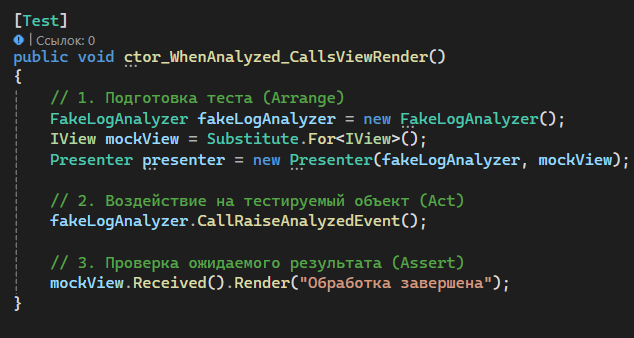
1. Создаём поддельный объект в файле PresenterTests.cs.



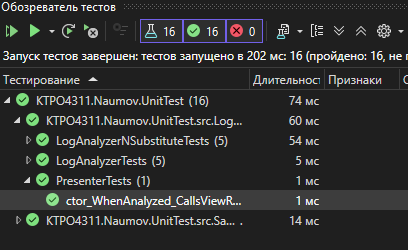
1. Создаём тестовый метод ctor\_WhenAnalyzed\_CallsViewRender()

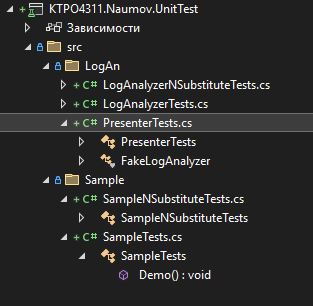


1. Реализуем тест:



1. Выполним тесты





1. **Имитация события с помощью NSubstitute**

**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.