Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем и программной инженерии

Лабораторная работа №4

по дисциплине "Технологии программирования"

ТЕМА РАБОТЫ:

Разработка Unit-тестов с помощью фреймворка XUnit

Выполнил:

студент гр. ПРИ-120

Парахин К.В.

Принял:

Ассистент кафедры ИСПИ

Данилов В.В.

Владимир 2022 г.

Цель работы:

Изучить возможности Unit тестирования .NET приложений с использованием framework XUnit

Выполнение работы:

Разработка через тестирование (TDD) — техника разработки программного обеспечения, которая основывается на повторении очень коротких циклов разработки: сначала пишется тест, покрывающий желаемое изменение, затем пишется код, который позволит пройти тест, и под конец проводится рефакторинг нового кода к соответствующим стандартам.

Установим необходимые пакеты NuGet для решения, чтобы можно было выполнять тесты Xunit из обозревателя тестов:

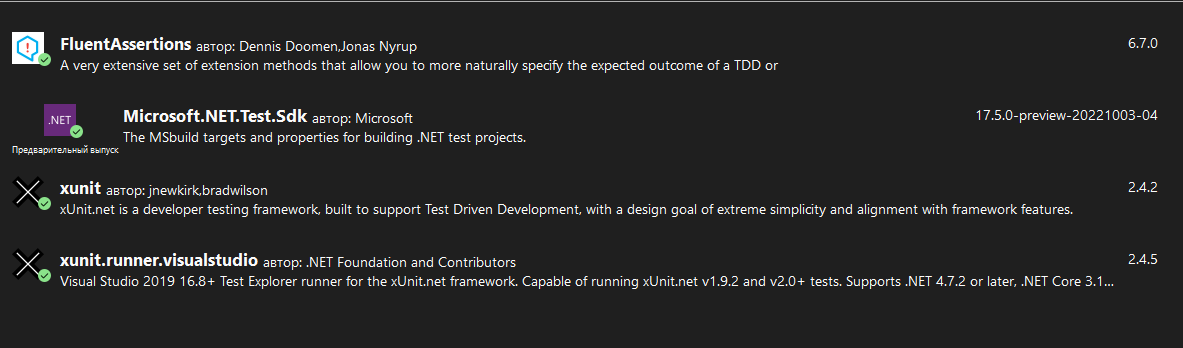


Рисунок 1. Установка пакетов NuGet

Начнем работу с реализации примера из методического указания:

Для начала напишем код для первого теста:

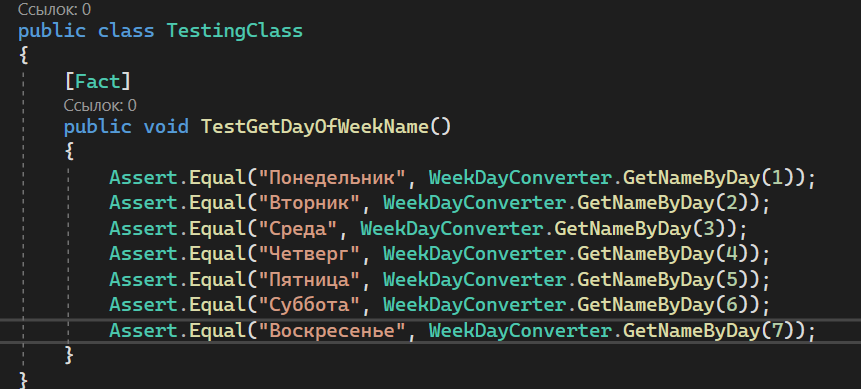
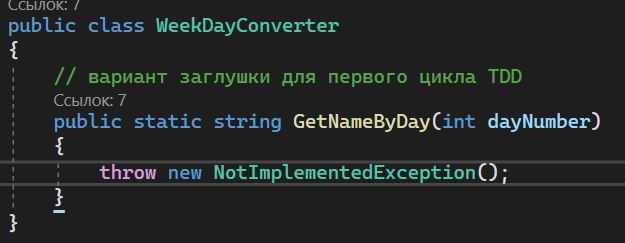
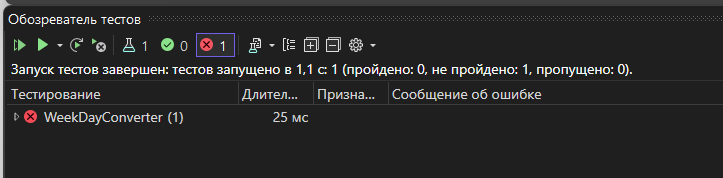


Рисунок 2. Код для тестового метода TestGetDayOfWeekName

Затем на 1 цикле TDD пишем для него заглушку в основном классе WeekDayConverter, чтобы программа могла скомпилироваться и тест запускался.

Но тест будет падать, так как логика для его выполнения пока не была реализована





Рисунки 3.1 - 3.2 – Результат выполнения теста на 1 цикле TDD

На 2 цикле TDD в основном классе WeekDayConverter реализуем логику выполнения тестового метода (через оператор множественного выбора switch):

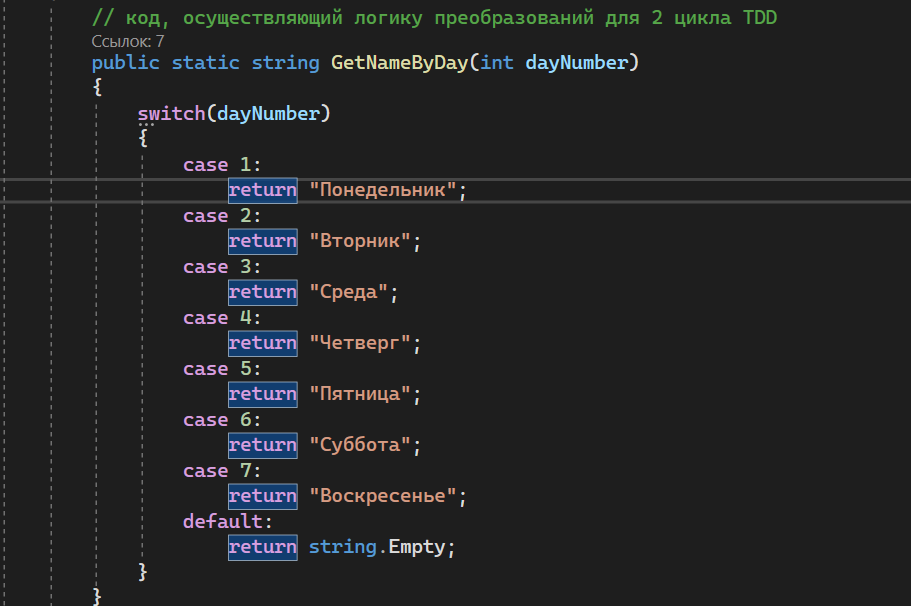


Рисунок 4. Код для второго цикла TDD

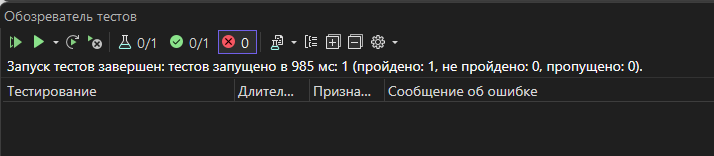


Рисунок 5. Успешное выполнение теста на 2 цикле TDD

Затем на 3 цикле TDD проводим рефакторинг написанного кода, чтобы ускорить его выполнение и получаем следующую картину:

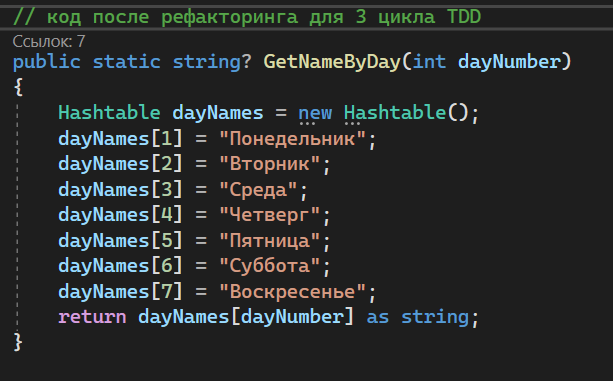


Рисунок 6. Код после рефакторинга для 3 цикла TDD

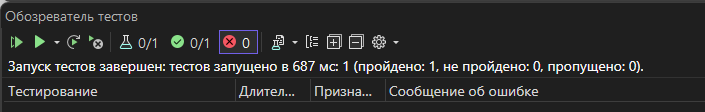


Рисунок 7. Успешное выполнение теста на 3 цикле TDD

Затем напишем дополнительный тестовый метод, проверяющий, будет ли выброшено исключение ArgumentException при попытке передать в метод WeekDayConverter.GetNameByDay() аргумента 8 (то есть дня недели, которого не существует и на который не рассчитана бизнесс-логика)

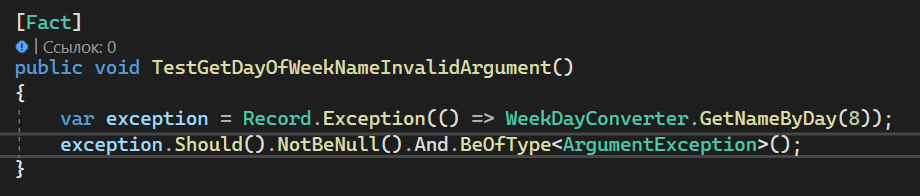


Рисунок 8. Код дополнительного тестового метода

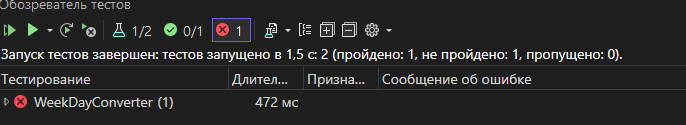


Рисунок 9. Результат выполнения тестов (новый тестовый метод падает)

Далее реализуем в методе GetNameByDay() класса WeekDayConverter логику проверки возвращаемого результата. Тестовый метод будет успешно выполняться:

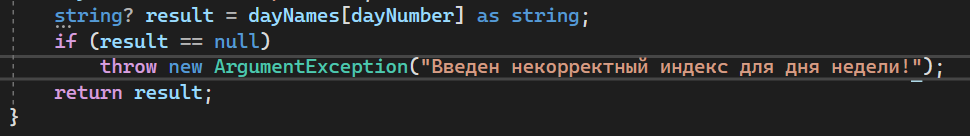


Рисунок 10. Добавление дополнительной логики в метод

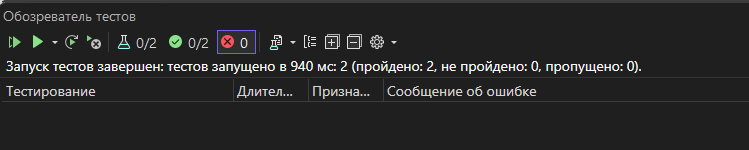


Рисунок 11. Успешное прохождение нового тестового метода

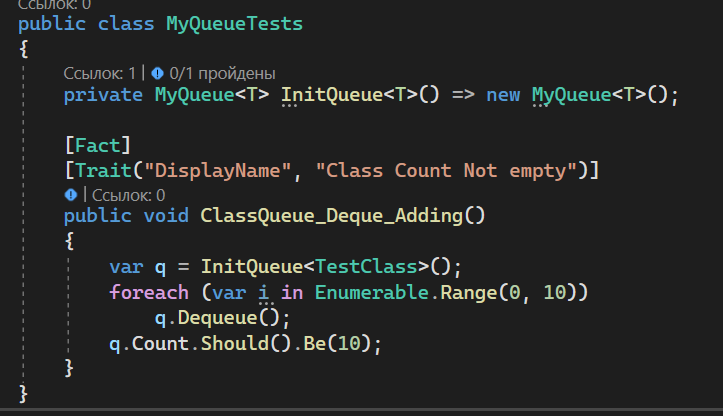
(цикл TDD для этого тестового метода звершен)

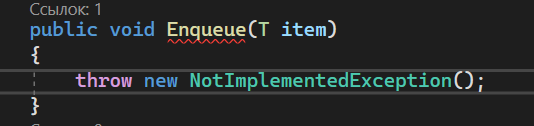
В данной работе я проведу примеры тестирования различных методов работы с классом очереди (Queue). Для этого я сам реализую обобщенный класс MyQueue<T> и напишу для него такие базовые методы, как: Dequeue, Enqueue, а также подсчет кол-ва элементов (Count) и базовый итератор с методами MoveNext, Current и Reset.

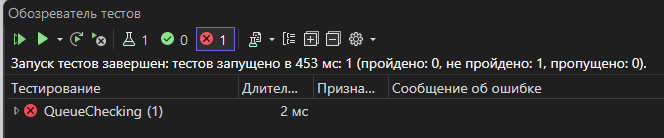
Далее приступим к написанию Unit-тестов (5 методов) по методологии цикла TDD:

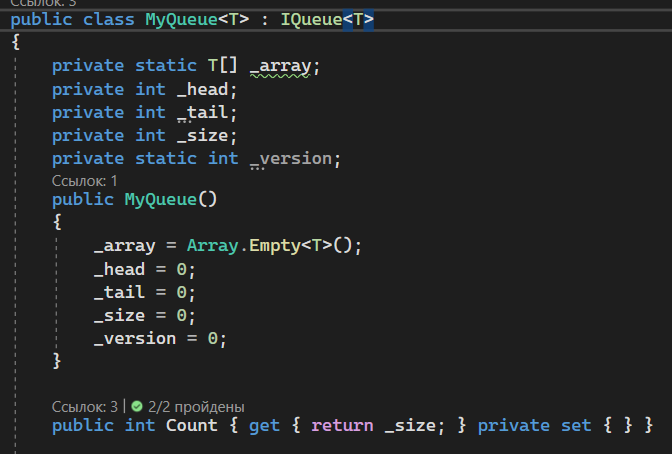
1. Метод, в котором мы попытаемся вызвать метод Enqueue (который берет и добавляет элемент в конец очереди) для пустой очереди.

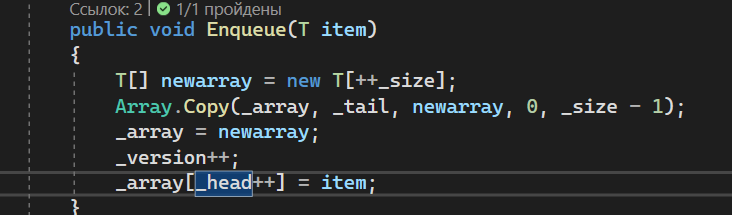
На 1 этапе мы напишем вот такой код тестового метода:

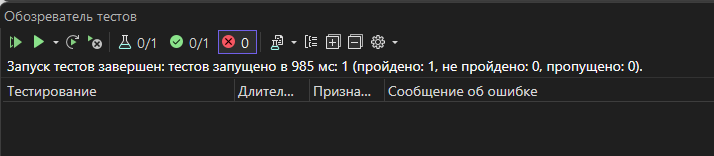




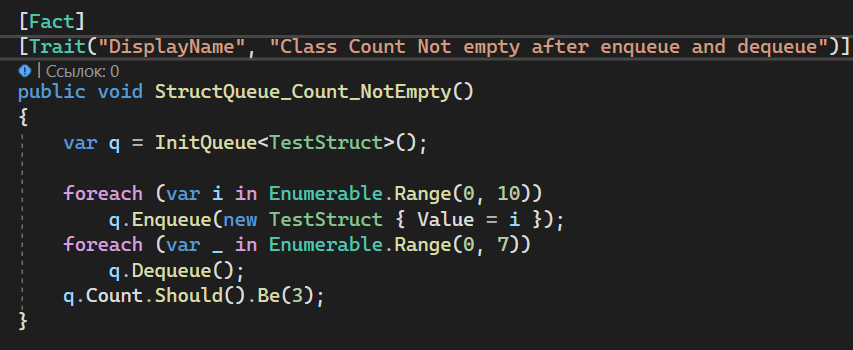


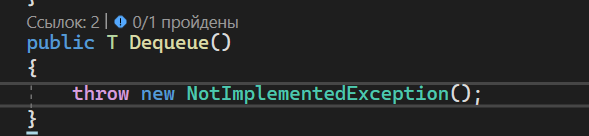


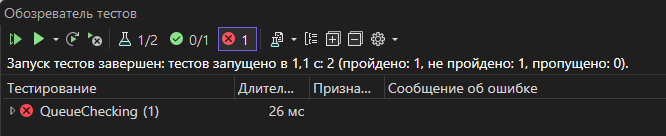


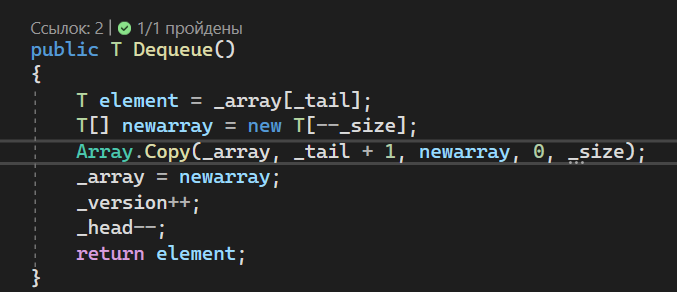


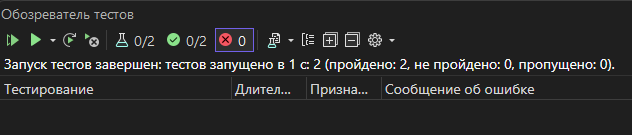
1. Метод, проверяющий вызов методов Enqueue и Dequeue (который берет и удаляет элемент из начала очереди в цикле):



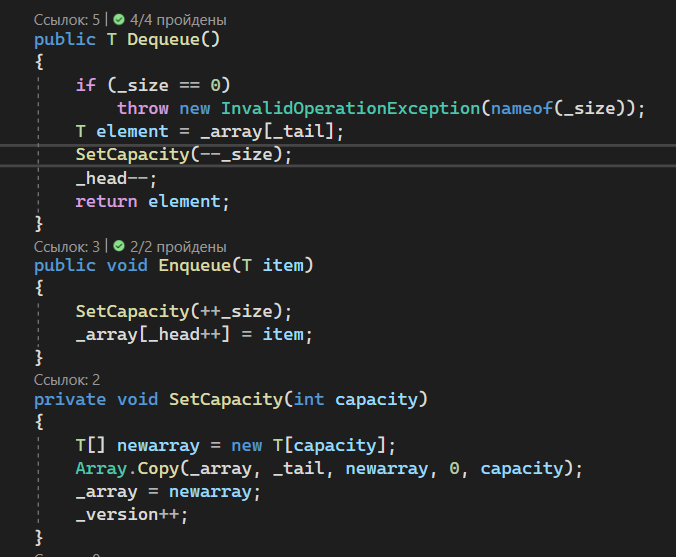


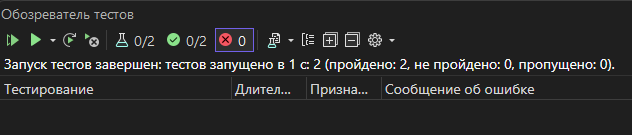




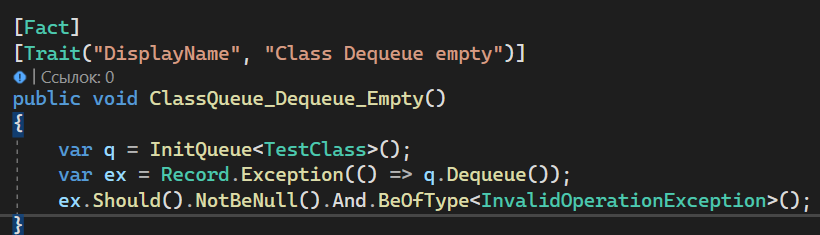


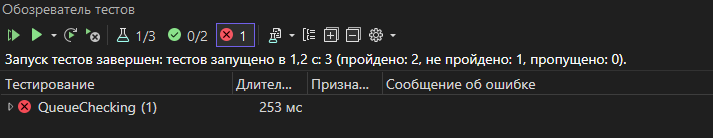
Рефакторинг методов Enqueue() и Dequeue() (3 цикл TDD):

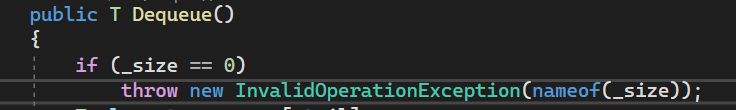


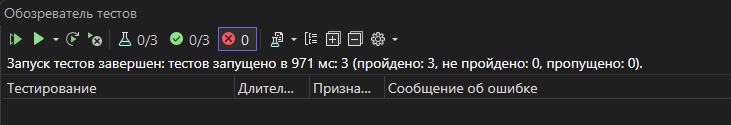


1. Метод, проверяющий вызов методов Dequeue для пустой очереди:

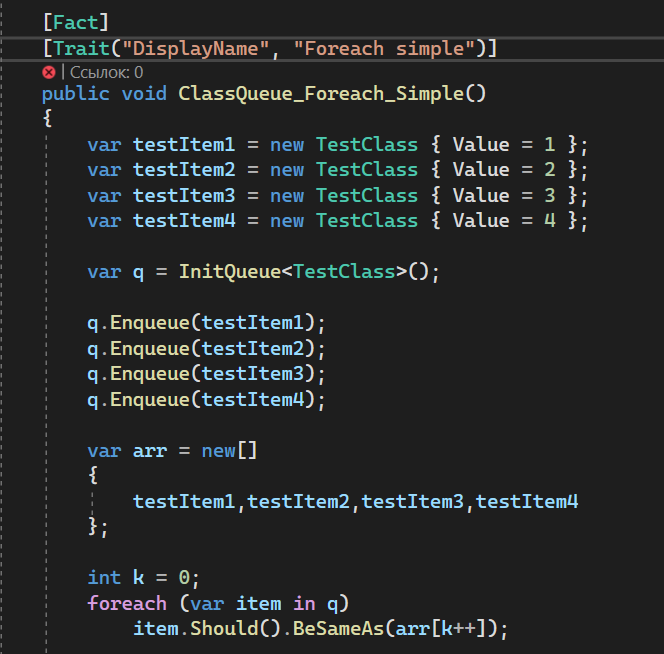


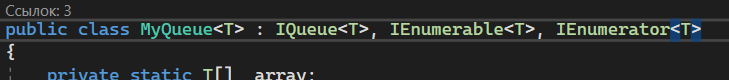


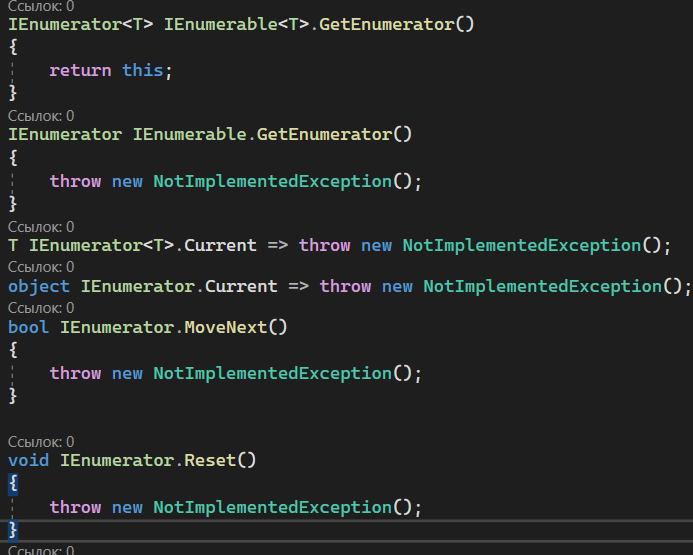


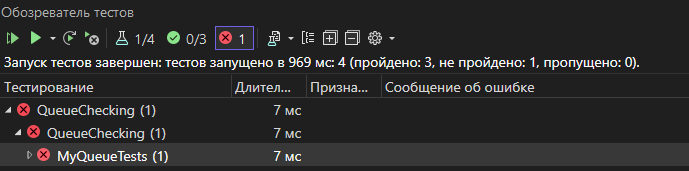


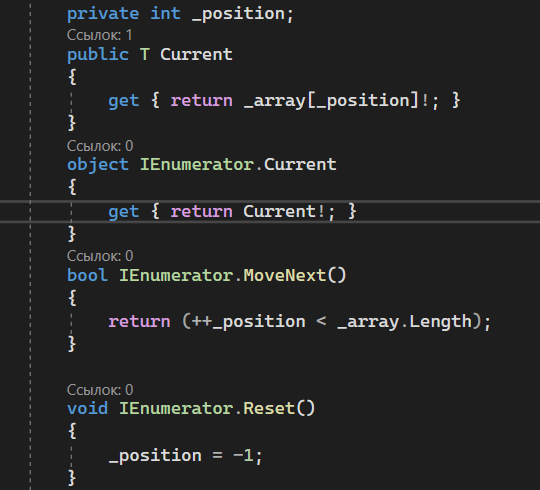
1. Метод, проверяющий перебор элементов очереди циклом foreach:

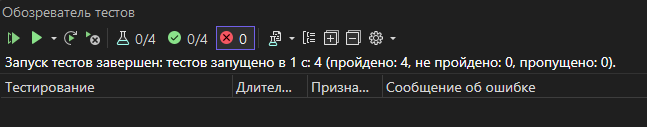




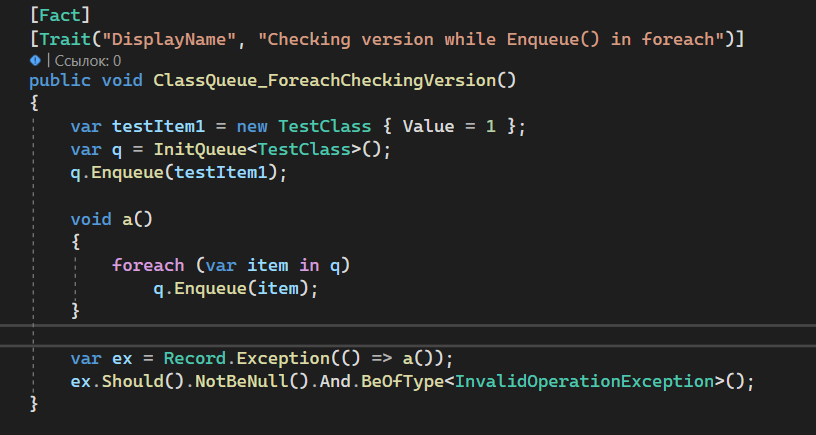


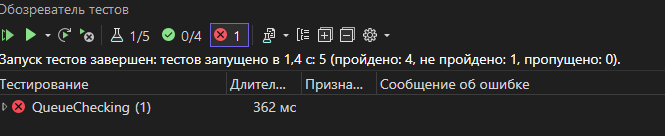


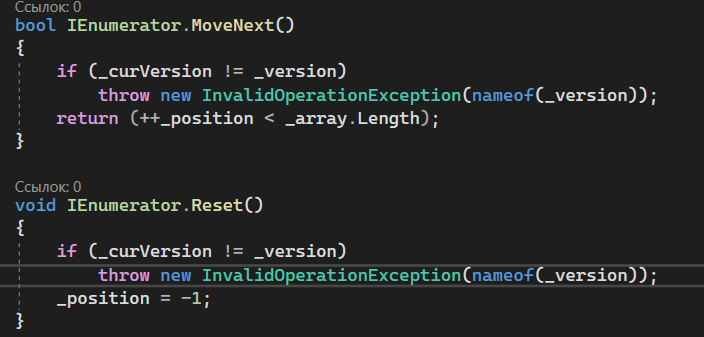


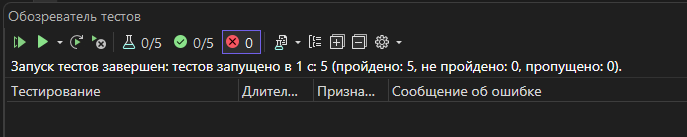


1. Метод, проверяющий возможность применения метода Enqueue в цикле foreach:









Вывод

В результате выполнения работы я изучил возможности Unit - тестирования .NET приложений с использованием framework Xunit, познакомился с использованием методологии TDD для итерационного создания тестов программной логики приложения.