МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федерального государственного бюджетного образовательного учреждения
высшего образования
«РОССИЙСКИЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ Г.В.ПЛЕХАНОВА»
Техникум Пермского института (филиала)

Практическая работа №1
по дисциплине «Поддержка и тестирование программных модулей»
по теме: Создание и запуск модульных тестов для управляемого кода

Работу выполнил:
Студент ИПо-21 Дерябин Максим Сергеевич

Пермь, 2026 г.

ОГЛАВЛЕНИЕ

ЦЕЛЬ РАБОТЫ

-Научиться создавать проекты модульных тестов в Visual Studio 2022

-Освоить использование атрибутов `[TestClass]` и `[TestMethod]`

-Научиться использовать методы проверки Assert

-Освоить работу с Test Explorer (Обозревателем тестов)

-Получить навыки поиска и исправления ошибок с помощью модульного тестирования

ХОД ВЫПОЛНЕНИЯ РАБОТЫ

1. Создание проекта Bank

Сначала я запустил Visual Studio 2022 и создал новый проект типа "Консольное приложение" на C# для .NET 6. Назвал проект Bank.

Затем я заменил содержимое файла Program.cs кодом класса BankAccount, который содержит:

- Приватные поля для хранения имени клиента и баланса
- Конструктор для инициализации счета
- Свойства только для чтения CustomerName и Balance
- Метод Debit для снятия денег со счета
- Метод Credit для пополнения счета

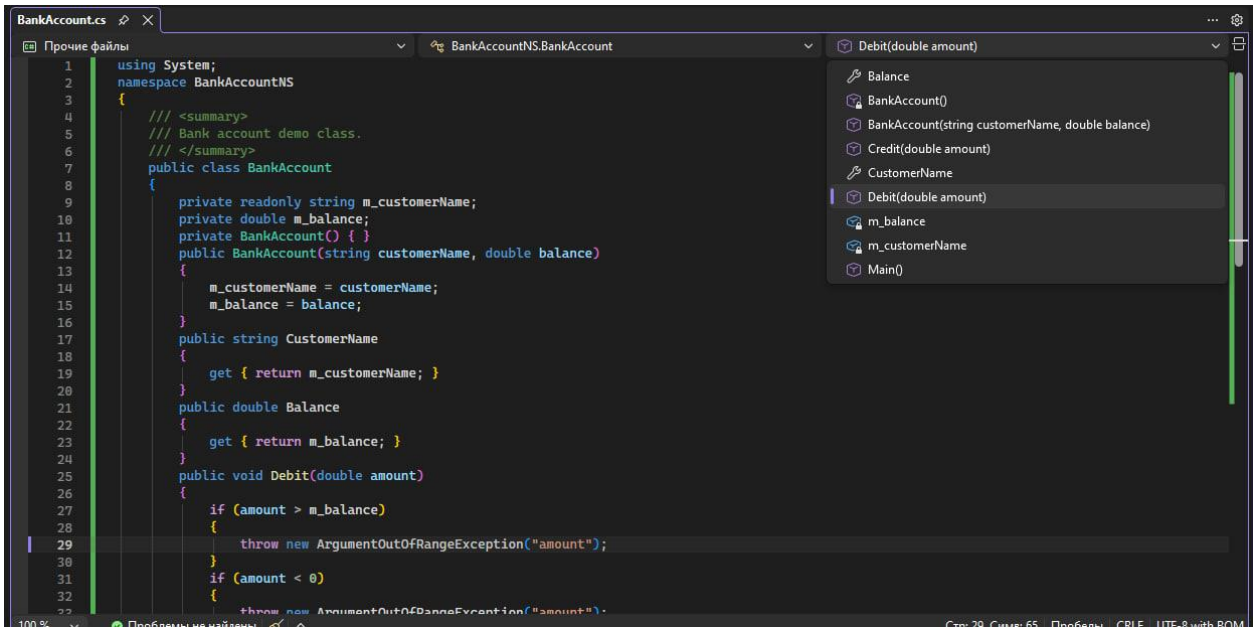

Рисунок 1.

После этого я переименовал файл Program.cs в BankAccount.cs и построил решение (Ctrl+Shift+B).

2. Создание проекта модульных тестов BankTests

Для создания тестов я добавил в решение новый проект типа "Проект модульного теста MSTest (.NET Core)" и назвал его BankTests.

Затем я добавил ссылку на проект Bank в проекте BankTests:

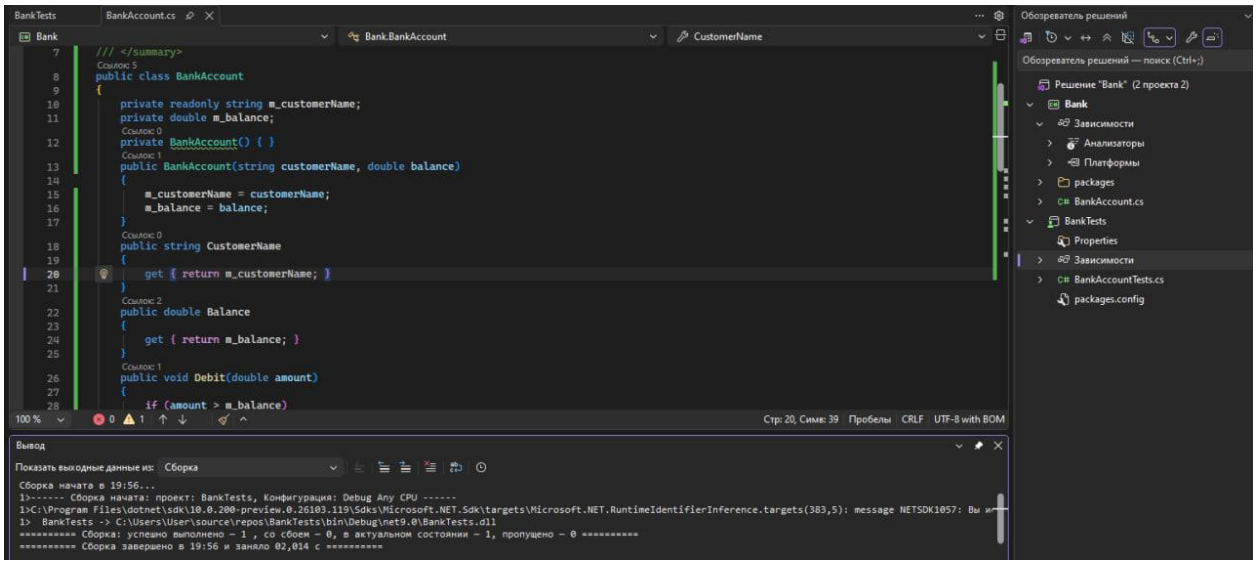

Рисунок 2.

Файл UnitTest1.cs я переименовал в BankAccountTests.cs, а класс UnitTest1 переименовал в BankAccountTests.

3. Написание первого теста

Первым я написал тест для проверки корректного списания средств со счета. Метод получил название `Debit_WithValidAmount_UpdatesBalance`.

Тест выполняет следующие действия:

- Arrange: Создает объект BankAccount с начальным балансом 11.99
- Act: Вызывает метод Debit со суммой 4.55
- Assert: Проверяет, что баланс стал равен 7.44

```
[TestMethod]
public void Debit_WithValidAmount_UpdatesBalance()
{
    // Arrange
    double beginningBalance = 11.99;
```

```
double debitAmount = 4.55;
double expected = 7.44;
BankAccount account = new BankAccount("Mr. Bryan Walton", beginningBalance);

// Act
account.Debit(debitAmount);

// Assert
double actual = account.Balance;
Assert.AreEqual(expected, actual, 0.001, "Account not debited correctly");
}
```

4. Запуск тестов и исправление ошибок

Я открыл Обозреватель тестов (Ctrl+E, T) и запустил все тесты (Ctrl+R, V).

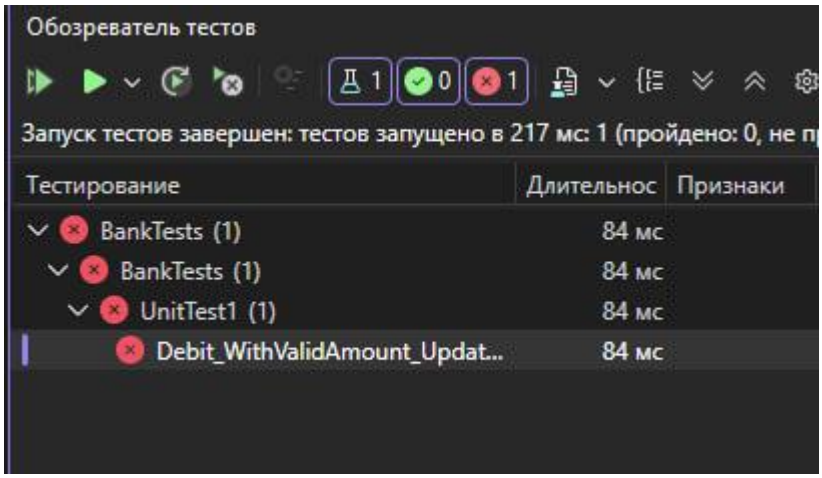

Рисунок 3.

Тест не прошел! Оказалось, что в методе Debit была ошибка: вместо вычитания суммы из баланса происходило ее добавление:

// Было (ошибочно):

m_balance += amount;

// Стало (правильно):

m_balance -= amount;

После исправления кода я снова запустил тест, и на этот раз он прошел успешно.

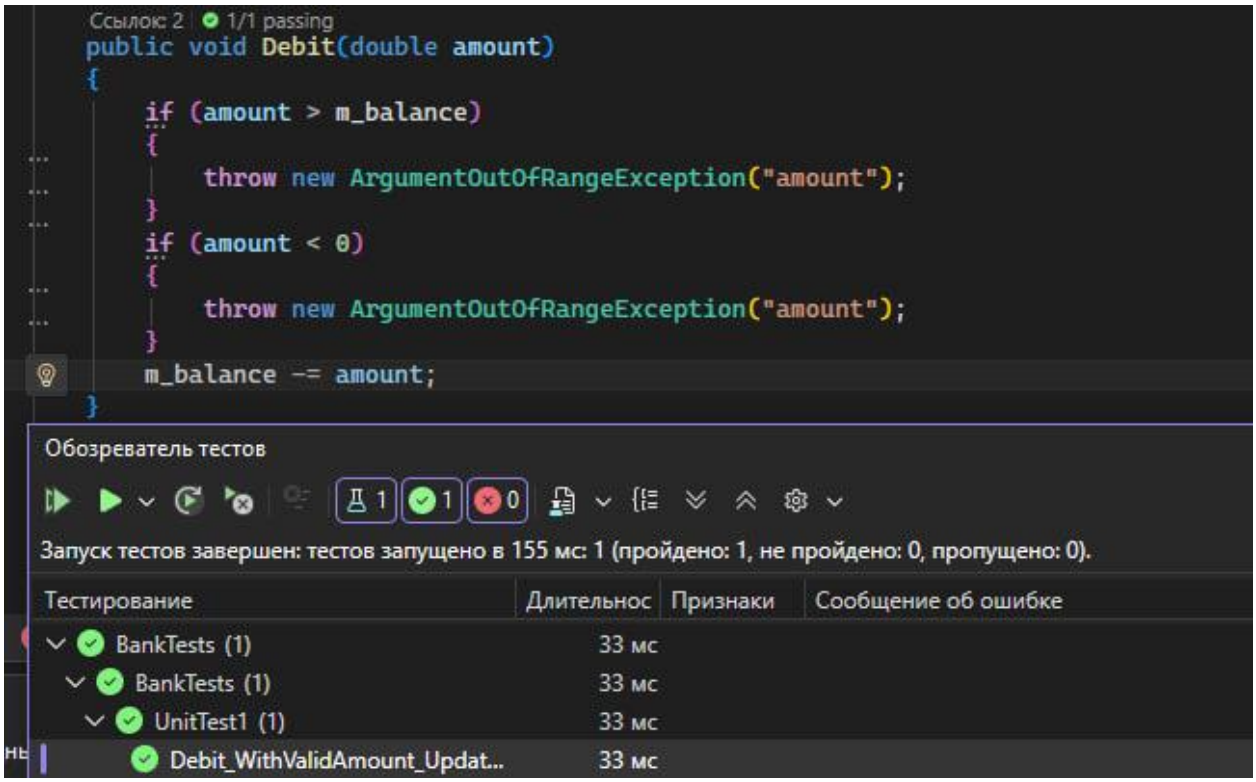

Рисунок 4.

5. Добавление тестов для проверки исключений

Далее добавил тесты для проверки того, что метод Debit выбрасывает исключение `ArgumentOutOfRangeException` в двух случаях:

Тест 1: Проверка исключения при отрицательной сумме списания

[TestMethod]

public void Debit_WhenAmountIsLessThanZero_ShouldThrowArgumentOutOfRange()

{

```
        // Arrange
        double beginningBalance = 11.99;
        double debitAmount = -100.00;
        BankAccount account = new BankAccount("Mr. Bryan Walton",
beginningBalance);

        // Act and assert
        Assert.ThrowsException<System.ArgumentOutOfRangeException>(()
=>
            account.Debit(debitAmount));
    }
```

Тест 2: Проверка исключения при сумме, превышающей баланс

```
    [TestMethod]
    public                                                    void
Debit_WhenAmountIsMoreThanBalance_ShouldThrowArgumentOutOfRange()
    {
        // Arrange
        double beginningBalance = 11.99;
        double debitAmount = 20.0;
        BankAccount account = new BankAccount("Mr. Bryan Walton",
beginningBalance);
        // Act
        try
        {
            account.Debit(debitAmount);
        }
        catch (System.ArgumentOutOfRangeException e)
        {
            // Assert
```

```
            StringAssert.Contains(e.Message,
BankAccount.DebitAmountExceedsBalanceMessage);
            return;
        }
        Assert.Fail("Expected exception not thrown");
    }
```

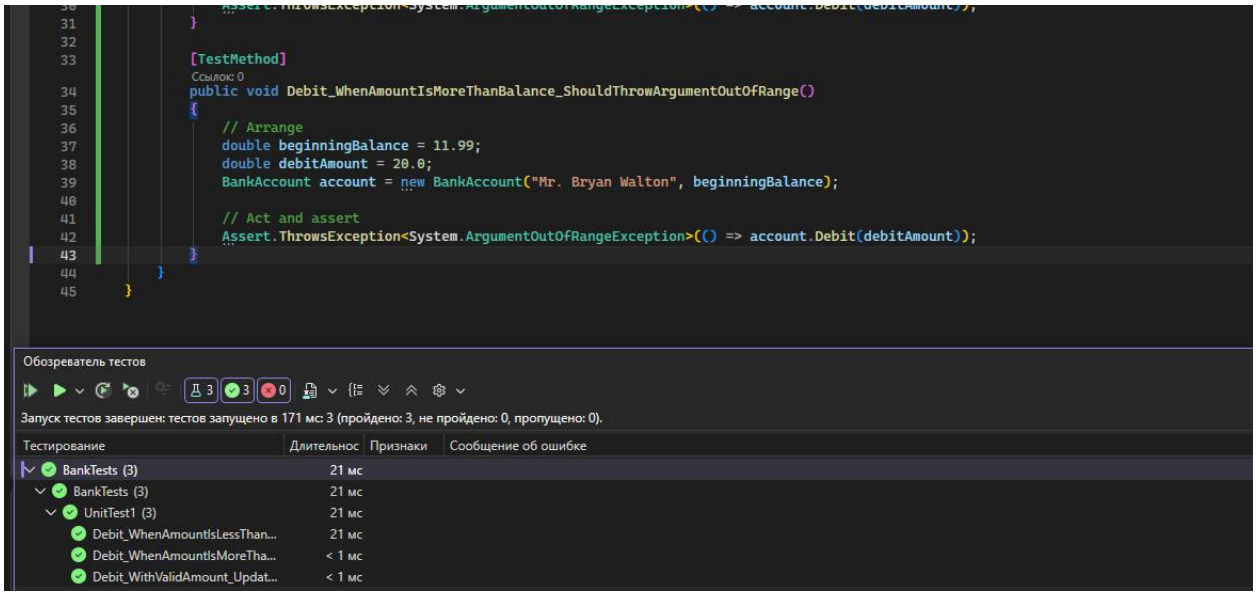

Рисунок 5.

После запуска всех трех тестов убедился, что все они проходят успешно.

РЕЗУЛЬТАТЫ РАБОТЫ

В ходе выполнения практической работы я:

1. Создал проект Bank с классом BankAccount

2. Создал проект модульных тестов BankTests

3. Настроил ссылку между проектами

4. Написал три тестовых метода:

`Debit_WithValidAmount_UpdatesBalance` - проверка корректного списания

`Debit_WhenAmountIsLessThanZero_ShouldThrowArgumentOutOfRange` - проверка исключения при отрицательной сумме

`Debit_WhenAmountIsMoreThanBalance_ShouldThrowArgumentOutOfRange` - проверка исключения при превышении баланса

5. Обнаружил и исправил ошибку в коде (+= вместо -=)

6. Убедился, что все тесты проходят успешно

ЗАКЛЮЧЕНИЕ

В результате выполнения данной практической работы я научился:

- Создавать проекты модульных тестов в Visual Studio 2022 с использованием MSTest

- Использовать атрибуты `[TestClass]` для тестовых классов и `[TestMethod]` для тестовых методов

- Применять различные методы проверки: `Assert.AreEqual`, `Assert.ThrowsException`, `StringAssert.Contains`

- Работать с обозревателем тестов (Test Explorer): запускать тесты, анализировать результаты

- Использовать модульное тестирование для поиска и исправления ошибок в коде

- Писать тесты для проверки корректной обработки исключительных ситуаций

Модульное тестирование оказалось очень полезным инструментом, который помог мне быстро обнаружить ошибку в методе Debit. Без тестов эта ошибка могла бы остаться незамеченной и привести к некорректной работе программы.

Все требования практической работы выполнены. Исходный код проекта размещен на GitHub, данная сопроводительная записка сохранена в формате PDF.