

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

ВВЕДЕНИЕ	Error! Bookmark not defined.
ЦЕЛЬ РАБОТЫ	3
ХОД ВЫПОЛНЕНИЯ РАБОТЫ.....	4
1 Создание проекта Bank	4
2 Создание проекта модульных тестов BankTests	4
3 Написание первого теста	5
4 Запуск тестов и исправление ошибок	6
5 Добавление тестов для проверки исключений	7
РЕЗУЛЬТАТЫ РАБОТЫ.....	10
ЗАКЛЮЧЕНИЕ	11

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

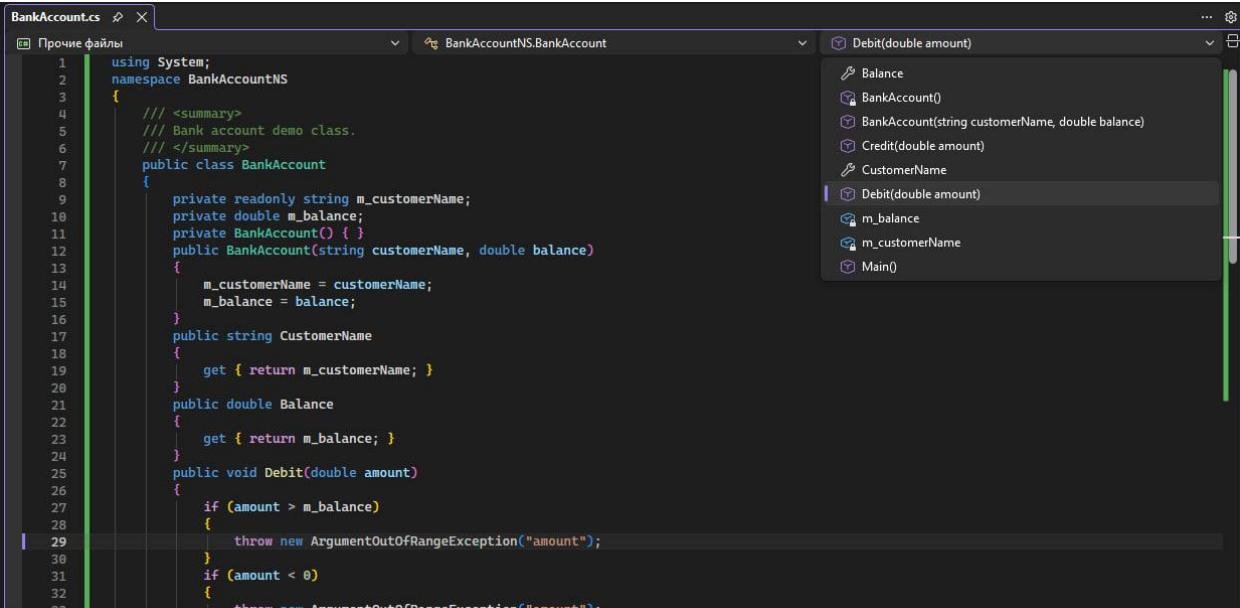
ХОД ВЫПОЛНЕНИЯ РАБОТЫ

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



```
BankAccount.cs > X
Прочие файлы  BankAccountNS.BankAccount
1  using System;
2  namespace BankAccountNS
3  {
4      /// <summary>
5      /// Bank account demo class.
6      /// </summary>
7      public class BankAccount
8      {
9          private readonly string _customerName;
10         private double _balance;
11         private BankAccount() { }
12         public BankAccount(string customerName, double balance)
13         {
14             _customerName = customerName;
15             _balance = balance;
16         }
17         public string CustomerName
18         {
19             get { return _customerName; }
20         }
21         public double Balance
22         {
23             get { return _balance; }
24         }
25         public void Debit(double amount)
26         {
27             if (amount > _balance)
28             {
29                 throw new ArgumentOutOfRangeException("amount");
30             }
31             if (amount < 0)
32             {
33                 throw new ArgumentOutOfRangeException("amount");
34             }
35             _balance -= amount;
36         }
37         public void Credit(double amount)
38         {
39             _balance += amount;
40         }
41     }
42 }
```

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:

The screenshot shows the Visual Studio IDE interface. In the center, there is a code editor window displaying the `BankAccount.cs` file from the `Bank` project. The code defines a `BankAccount` class with properties `CustomerName` and `Balance`, and a method `Debit`. The `CustomerName` property has a get accessor. In the bottom right corner of the code editor, there is a status bar with the text "Стр. 20, Симв. 39 Пробелы CRLF UTF-8 with BOM". To the right of the code editor is the "Solution Explorer" window, which shows two projects: `Bank` and `BankTests`. The `Bank` project contains files like `Bank.cs`, `BankAccount.cs`, and `BankAccountTests.cs`. The `BankTests` project contains files like `BankAccountTests.cs` and `packages.config`. At the bottom of the screen, there is a "Output" window showing build logs. The log starts with "Сборка началась в 19:56..." and ends with "Сборка успешно завершена в 19:56 и заняло 02,014 с ****".

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).

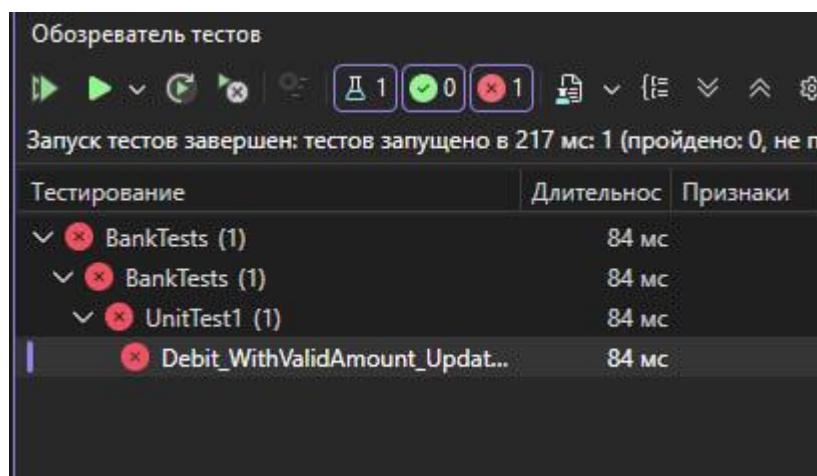


Рисунок 3.

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.

The screenshot shows the Visual Studio IDE interface. At the top, there is a code editor window with the following C# code:

```

Ссылка 2 1/1 passing
public void Debit(double amount)
{
    if (amount > m_balance)
    {
        throw new ArgumentOutOfRangeException("amount");
    }
    if (amount < 0)
    {
        throw new ArgumentOutOfRangeException("amount");
    }
    m_balance -= amount;
}

```

Below the code editor is the Test Explorer window titled "Обозреватель тестов". It displays the following information:

- Test results: 1/1 passing (1 passed, 0 failed, 0 skipped).
- Summary: Запуск тестов завершен: тестов запущено в 155 мс: 1 (пройдено: 1, не пройдено: 0, пропущено: 0).
- Test list:

Тестирование	Длительность	Признаки	Сообщение об ошибке
BankTests (1)	33 мс		
BankTests (1)	33 мс		
UnitTest1 (1)	33 мс		
Debit_WithValidAmount_Updater...	33 мс		

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_ShouldThrowArgumentOutOfRangeException()

{

```

// 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_ShouldThrowArgumentOutOfRangeException()
{
    // 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");
}

```

The screenshot shows a code editor with C# test code and a Test Explorer window below it.

```

30     Assert.ThrowsException<System.ArgumentOutOfRangeException>(() => account.Debit(debitAmount));
31 }
32
33 [TestMethod]
34 public void Debit_WhenAmountIsMoreThanBalance_ShouldThrowArgumentOutOfRangeException()
35 {
36     // Arrange
37     double beginningBalance = 11.99;
38     double debitAmount = 20.0;
39     BankAccount account = new BankAccount("Mr. Bryan Walton", beginningBalance);
40
41     // Act and assert
42     Assert.ThrowsException<System.ArgumentOutOfRangeException>(() => account.Debit(debitAmount));
43 }
44
45 }

```

Обозреватель тестов

Запуск тестов завершен: тестов запущено в 171 мс: 3 (пройдено: 3, не пройдено: 0, пропущено: 0).

Тестирование	Длительность	Признаки	Сообщение об ошибке
BankTests (3)	21 мс		
BankTests (3)	21 мс		
UnitTest1 (3)	21 мс		
Debit_WhenAmountIsLessThan...	21 мс		
Debit_WhenAmountIsMoreTha...	< 1 мс		
Debit_WithValidAmount_Uptad...	< 1 мс		

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_ShouldThrowArgumentOutOfRangeException`
- проверка исключения при отрицательной сумме
`Debit_WhenAmountIsMoreThanBalance_ShouldThrowArgumentOutOfRangeException`
- 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.