

**Пермский институт (филиал) федерального
государственного бюджетного образовательного
учреждения высшего образования
«Российский экономический университет имени Г.В.
Плеханова»**

Кафедра информационных технологий и программирования.

Сопроводительная записка к практической работе №1.
Тема работы: «Средства тестирования Visual Studio-2022».

Работу выполнила:
Александрович Валерия
Николаевна
Группа: ИПс-11
Преподаватель: Берестов
Дмитрий Борисович

Пермь 2026

Оглавление.

Введение.....	3
1. Создание проекта для тестирования	4
2. Создание проекта модульного теста	11
3. Создание тестового класса.....	15
4. Создание метода теста.....	17
5. Сборка и запуск теста	19
6. Исправление кода и повторный запуск тестов	22
7. Создание и запуск новых методов теста	23
8. Рефакторинг test-кода.....	25
9. Рефакторинг тестовых методов.....	27
10. Повторное тестирование, переписывание и анализ	28
Заключение	30

Введение.

Выполняя практическую работу №1 по теме «Средства тестирования Visual Studio-2022», я ознакомилась и изучила учебное пособие Средства тестирования Visual Studio.

Практическая работа №1 была выполнена по руководству "Средства тестирования Visual Studio-2022", по стр. 158 -170.
(<https://cloud.mail.ru/public/JaXA/BUKbRzZoN>).

1. Создание проекта для тестирования.

1. Я запустила Visual Studio.

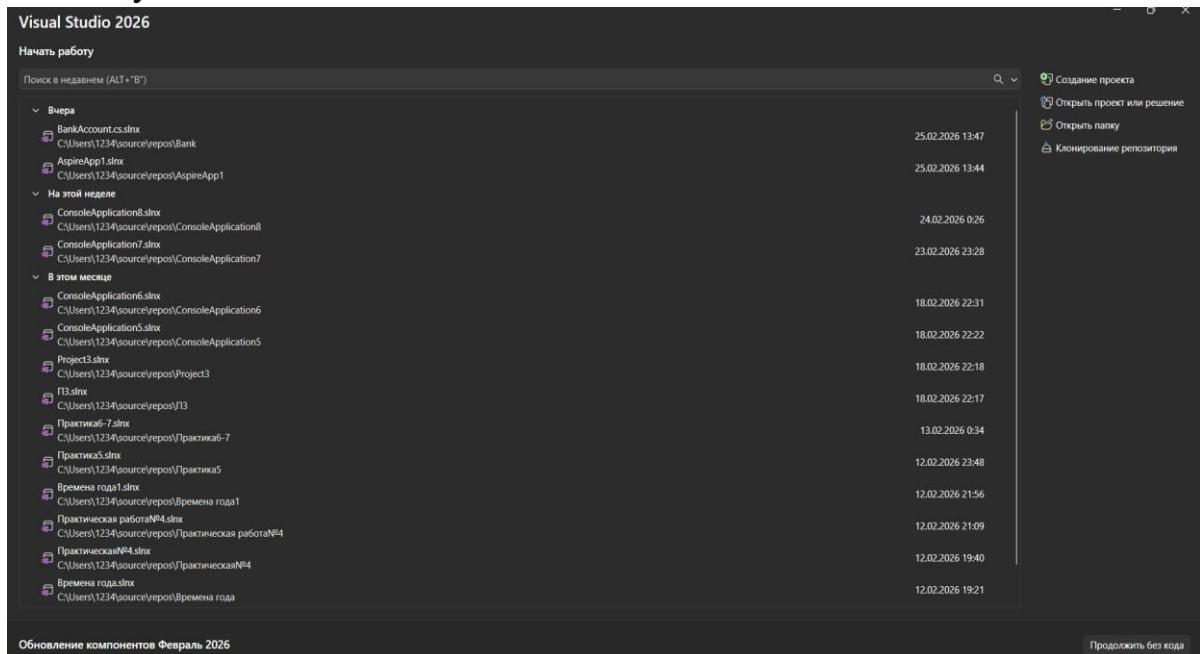


Рисунок 1

2. Я создала проект.

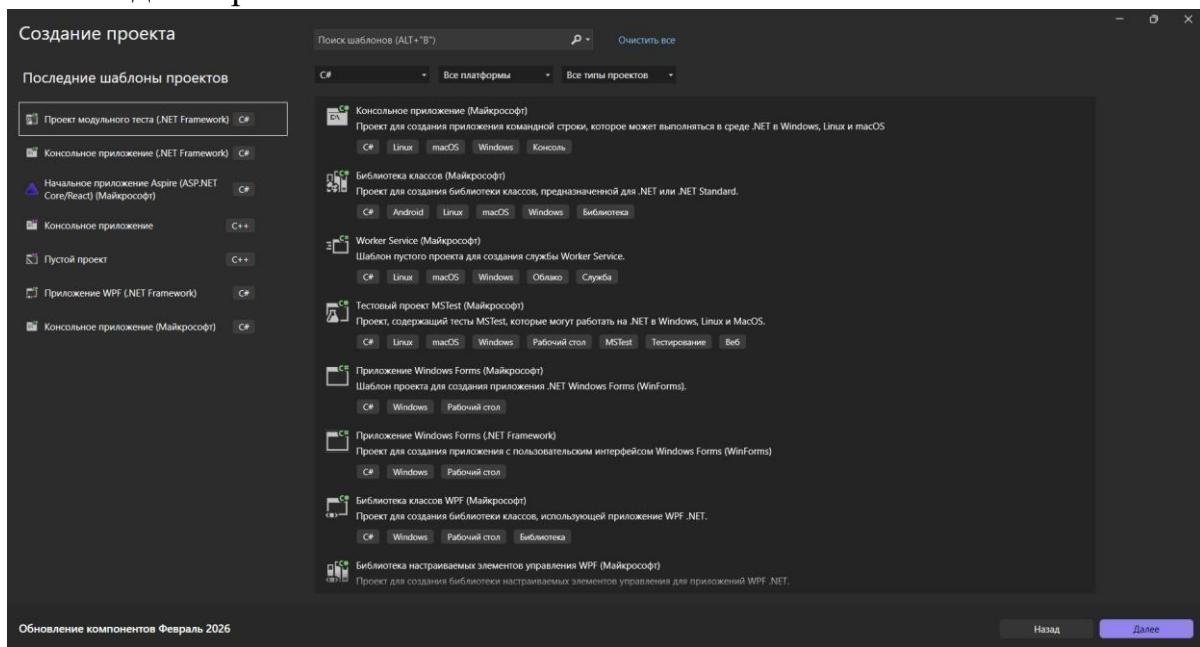


Рисунок 2

3. Я выбрала шаблон проекта Консольное приложение на C#.

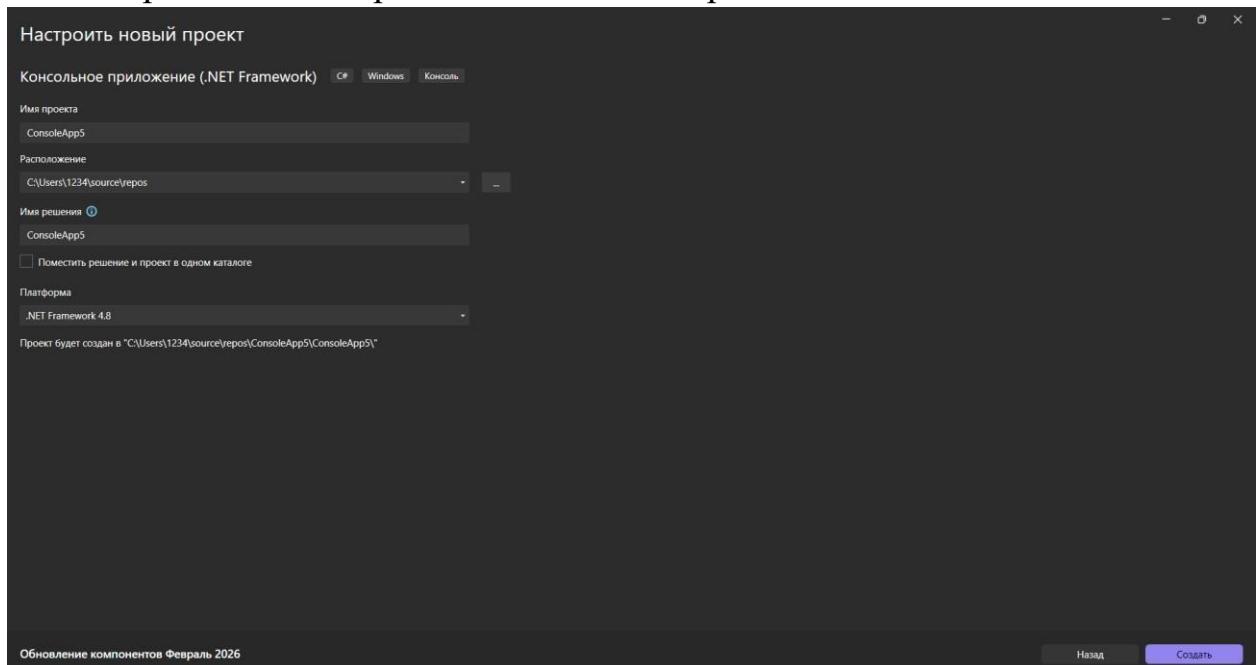


Рисунок 3

4. Я назвала проект Bank и выбрали рекомендуемую версию целевой платформы.

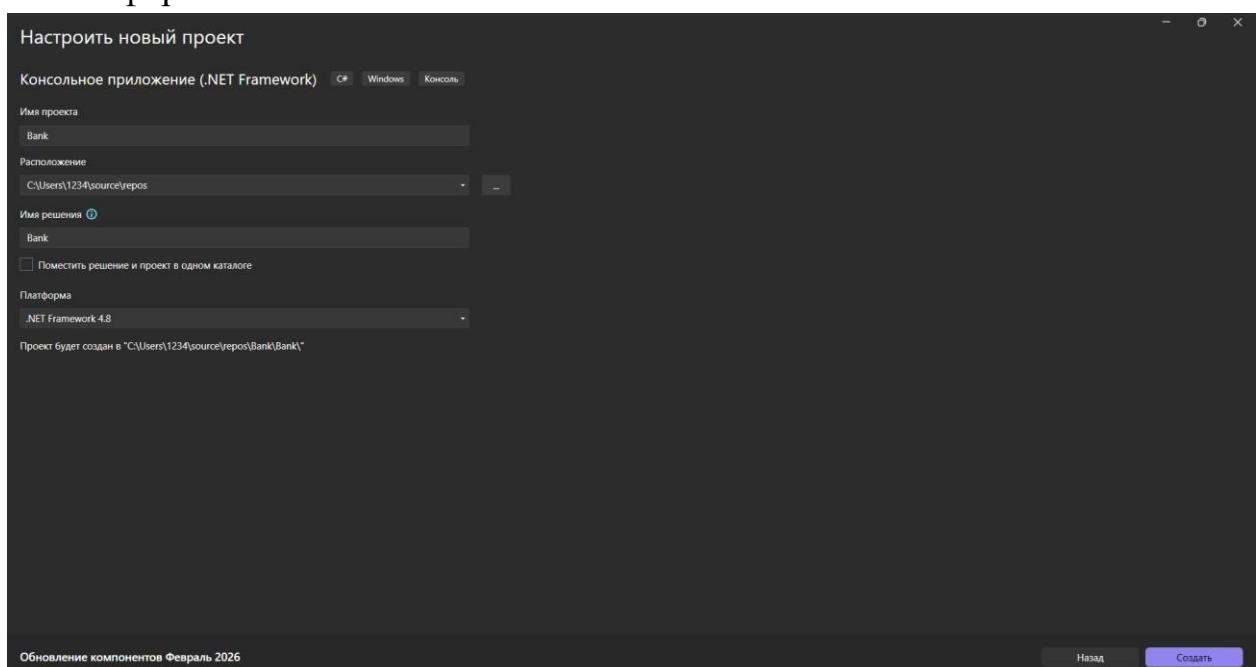


Рисунок 4

5. Я заменила содержимое файла Program.cs следующими кодом на C#, который определяет класс BankAccount.

Код:

```
using System;
namespace BankAccountNS
{
    /// <summary>
    /// Bank account demo class.
    /// </summary>    public
    class BankAccount
    {
        private readonly string m_customerName;
        private double m_balance;      private BankAccount() { }
        public BankAccount(string customerName, double balance)
        {
            m_customerName = customerName;
            m_balance = balance;
        }
        public string CustomerName
        {
            get { return m_customerName; }
        }
        public double Balance
        {
            get { return m_balance; }
        }
        public void Debit(double amount)
        {
            if (amount > m_balance)
            {
                throw new ArgumentOutOfRangeException("amount");
            }
            if (amount < 0)
            {
                throw new ArgumentOutOfRangeException("amount");
            }
            m_balance += amount; // intentionally incorrect code
        }
        public void Credit(double amount)
        {
            if (amount < 0)
            {
                throw new ArgumentOutOfRangeException("amount");
            }
        }
    }
}
```

```

        }
        m_balance += amount;
    }
    public static void Main()
    {
        BankAccount ba = new BankAccount("Mr. Bryan Walton",
            11.99);
        ba.Credit(5.77);
        ba.Debit(11.22);
        Console.WriteLine("Current balance
is ${0}", ba.Balance);
    }
}
}

```

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar says "Program.cs" and "Bank". The main window displays the C# code for the "Bank" project. The code defines a `Bank` namespace and a `BankAccount` class with properties `CustomerName` and `Balance`, and methods `Credit` and `Debit`. A `Main` method is shown creating a `BankAccount` instance and printing its balance. The status bar at the bottom indicates "Стр. 53, Симв. 6 Пробелы CRLF UTF-8 with BOM".

Рисунок 5

6. Я переименовала файл в BankAccount.cs.

Для этого я зашла в Обозреватель решений, нажала на Bank правой кнопки мыши и выбрала Переименовать.

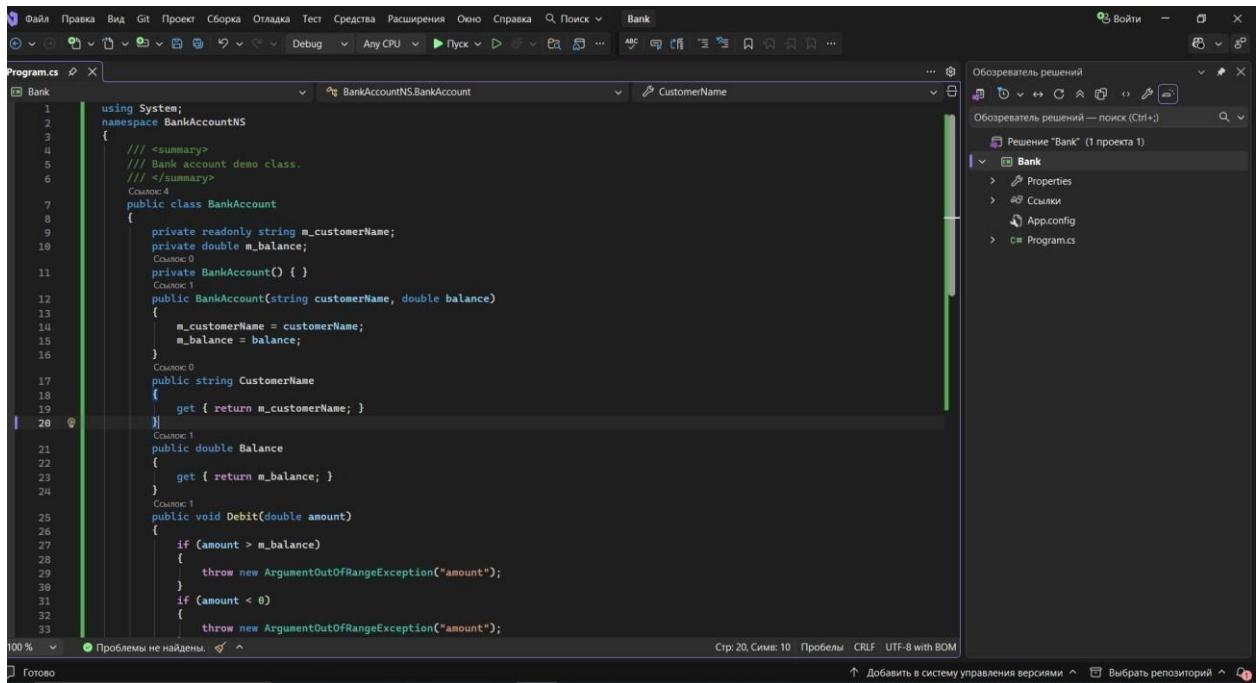


Рисунок 6

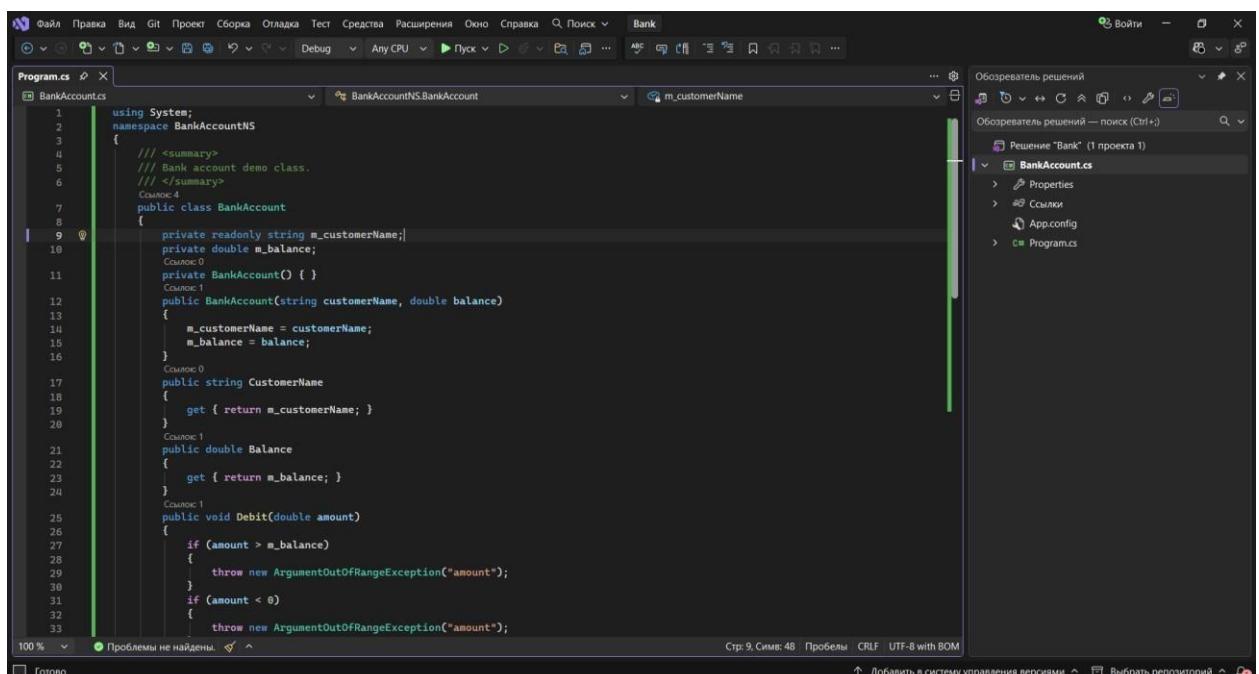


Рисунок 7

7. Я использовала клавиши CTRL +SHIFT + В для того, чтобы произошла сборка (вывод был).

The screenshot shows the Microsoft Visual Studio IDE interface. On the left, the code editor displays `BankAccount.cs` with C# code for a `BankAccount` class. On the right, the Solution Explorer shows a single project named "Bank" containing `BankAccount.cs`, `Properties`, `Sсылки`, `App.config`, and `Program.cs`. Below the code editor is the Output window, which shows the build logs:

```
Показать выходные данные из: Сборка
Сборка началась в 15:46 ...
1>----- Сборка начата: проект: BankAccount.cs, Конфигурация: Debug Any CPU -----
1> BankAccount.cs -> C:\Users\1234\source\repos\Bank\Bank\bin\Debug\Bank.exe
***** Сборка: успешно выполнено - 1 , со сбоями - 0, в актуальном состоянии - 0, пропущено - 0 *****
***** Сборка завершена в 15:46 и заняло 01,278 с *****
```

Рисунок 8

Результат моей работы:

The screenshot shows the Microsoft Visual Studio interface with the 'Bank' project open. The 'Program.cs' file is selected. The output window at the bottom displays the following text:

```
Показать выходные данные из: Сборка
Сборка началась в 14:24...
----- Сборка: успешно выполнено - 0 , со сбоем - 0, в актуальном состоянии - 1, пропущено - 0 -----
----- Сборка завершена в 14:24 в заняло 00:00:172 -----
```

Рисунок 9

The screenshot shows the Microsoft Visual Studio interface with the 'Bank' project open. The 'Program.cs' file is selected. The output window at the bottom displays the following text:

```
Current balance is $28,98
C:\Users\1123A\source\repos\Bank\Bank\b1n\Debug\Bank.exe (процесс 8892) завершил работу с кодом 0 (0x0).
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Берите" ->"Параметры" ->"Отладка" ->"Автоматически закрывать консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:
```

Рисунок 10

2. Создание проекта модульного теста.

1. В меню Файл я выбрала Добавить>Создать проект.

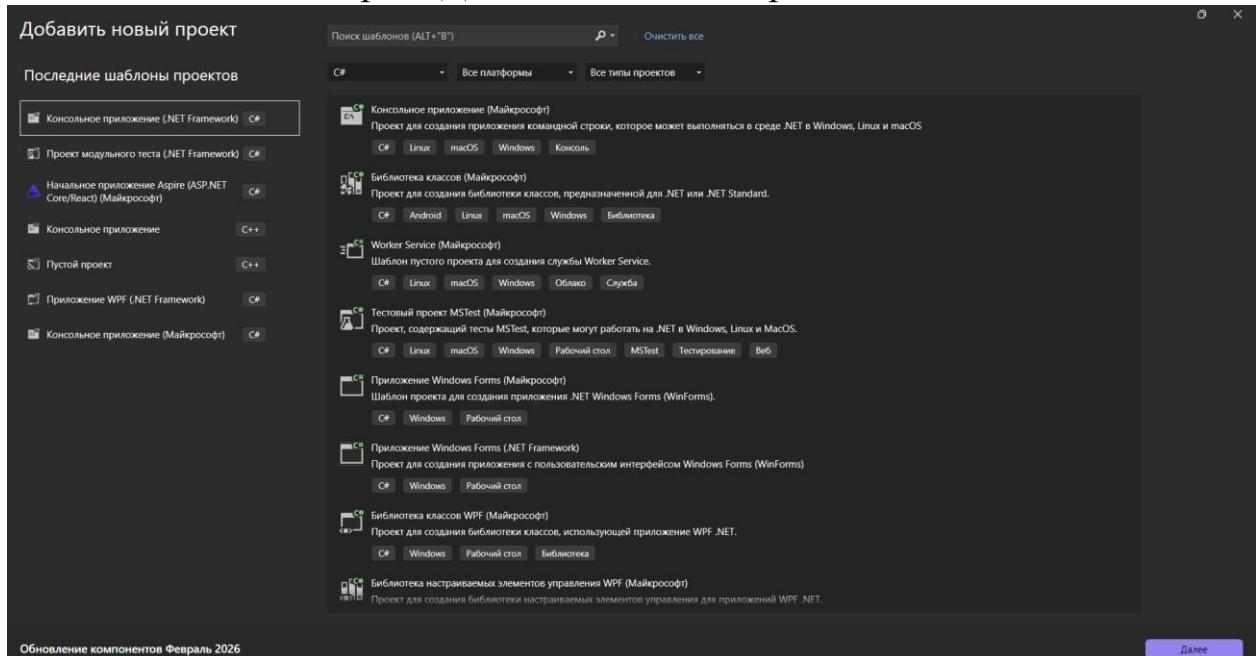


Рисунок 11

2. Я ввела test в поле поиска, выбрала C# в качестве языка, затем выбрала Проект модульного теста MSTest (.NET Core) для C# в качестве шаблона .NET Core.

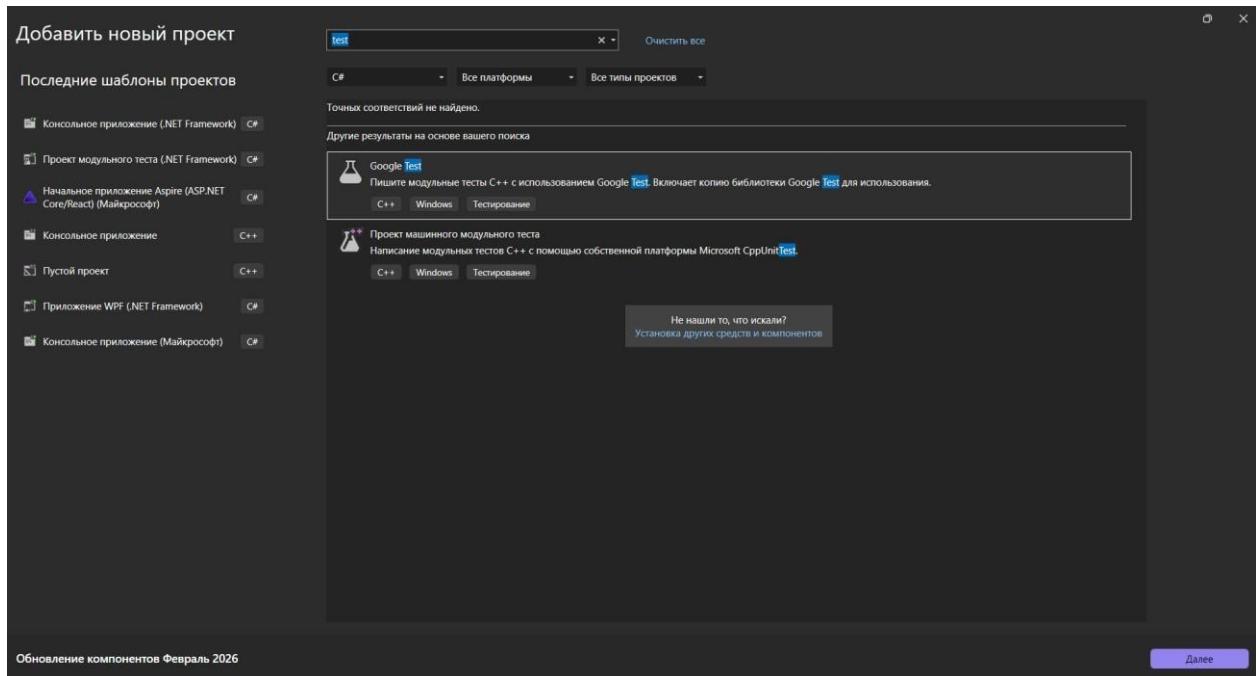


Рисунок 12

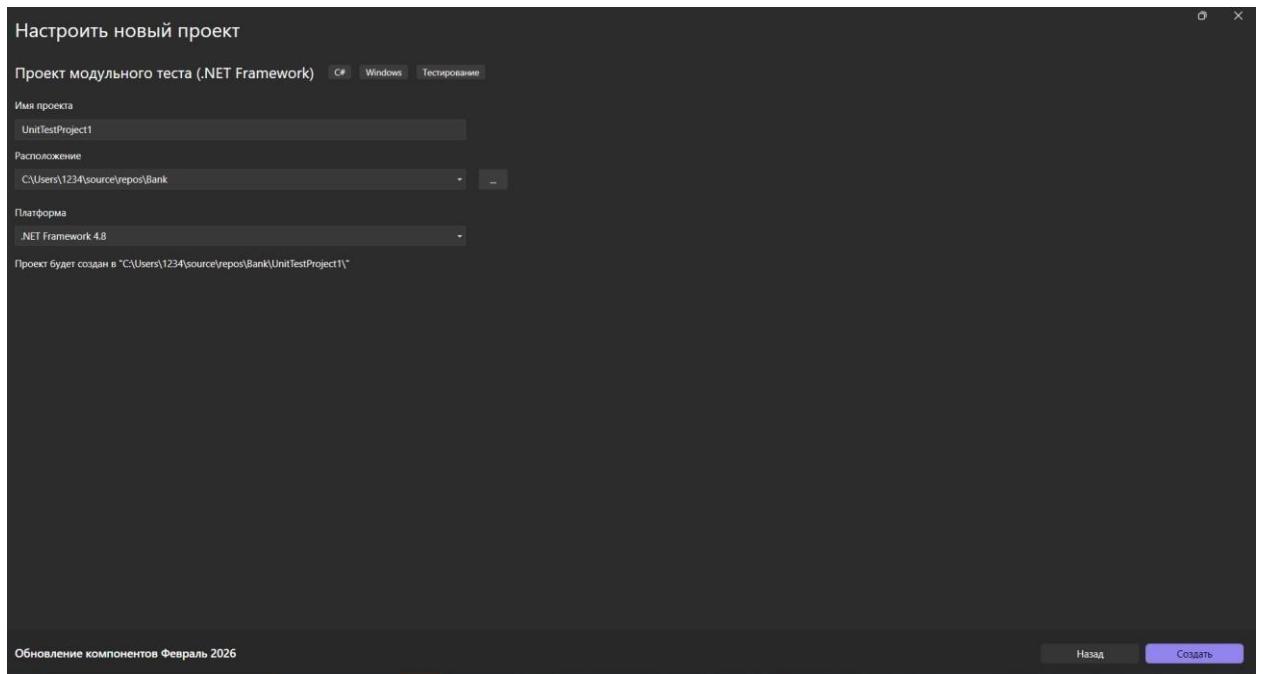


Рисунок 13

3. Я назвала проект BankTests.

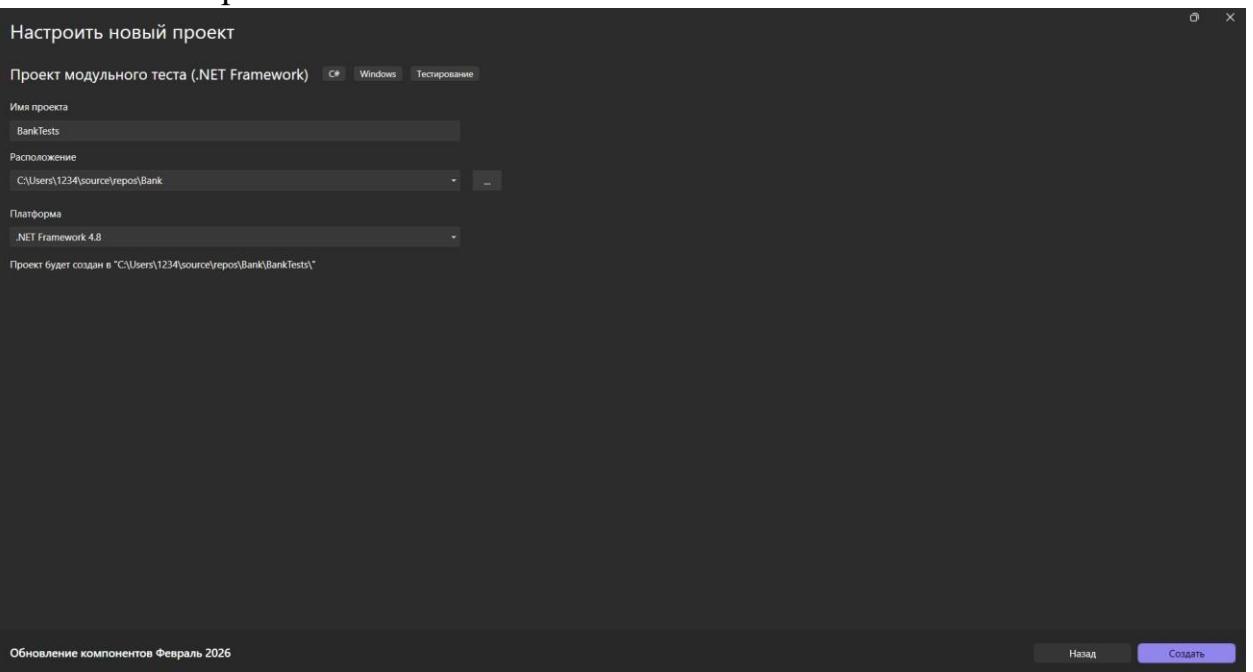


Рисунок 14

4. Я выбрала рекомендуемую версию целевой платформы.

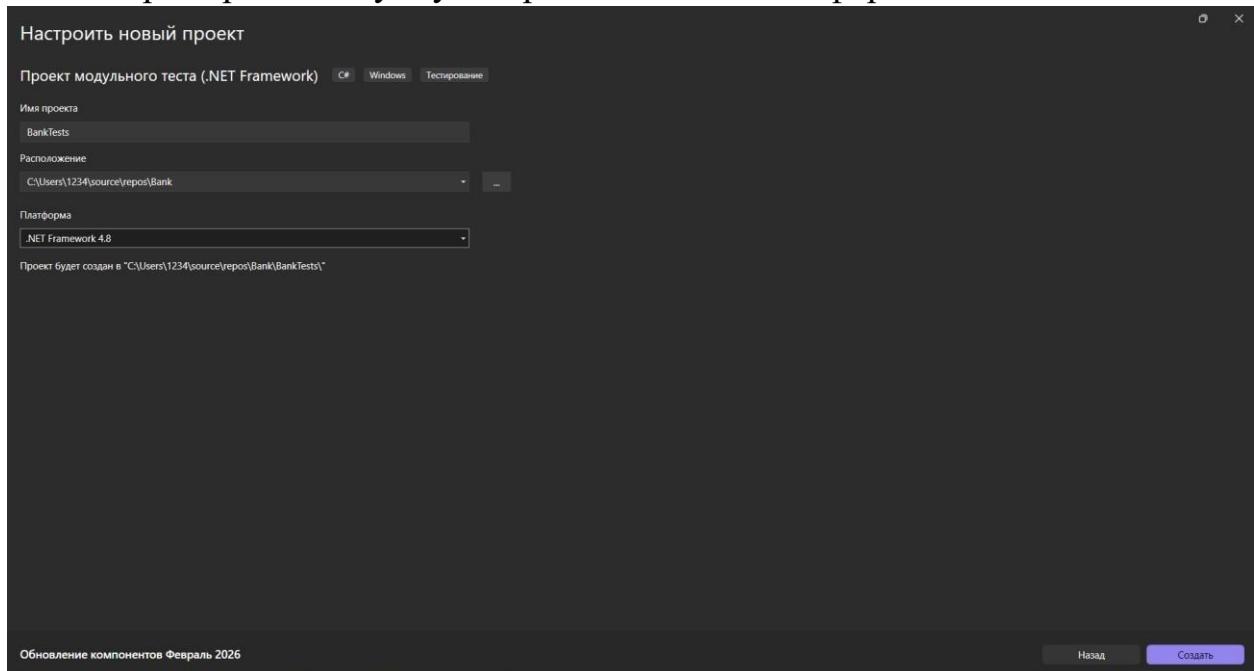


Рисунок 15

5. В проекте BankTests я добавила ссылку на проект Банк.

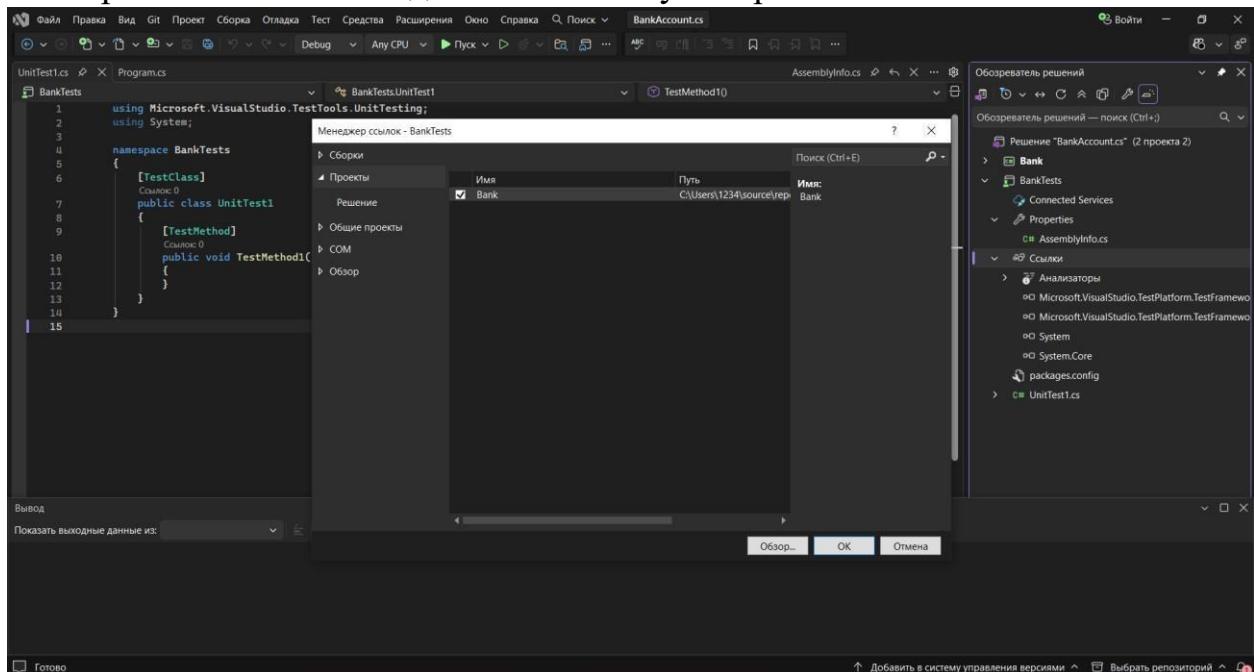


Рисунок 16

6. В диалоговом окне Диспетчер ссылок я развернула Проекты, выбрала Решение и выбрала элемент Банк и нажала кнопку ОК.

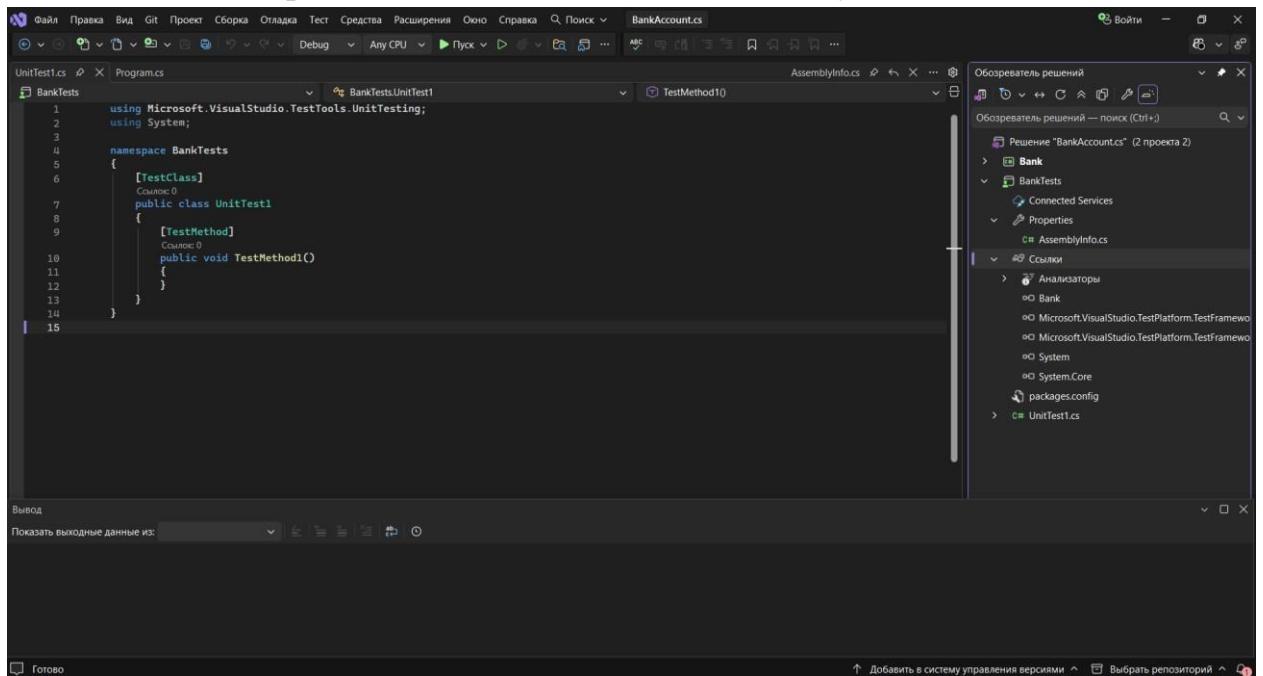


Рисунок 17

3. Создание тестового класса.

1. Я переименовала файл в BankAccountTests.cs.

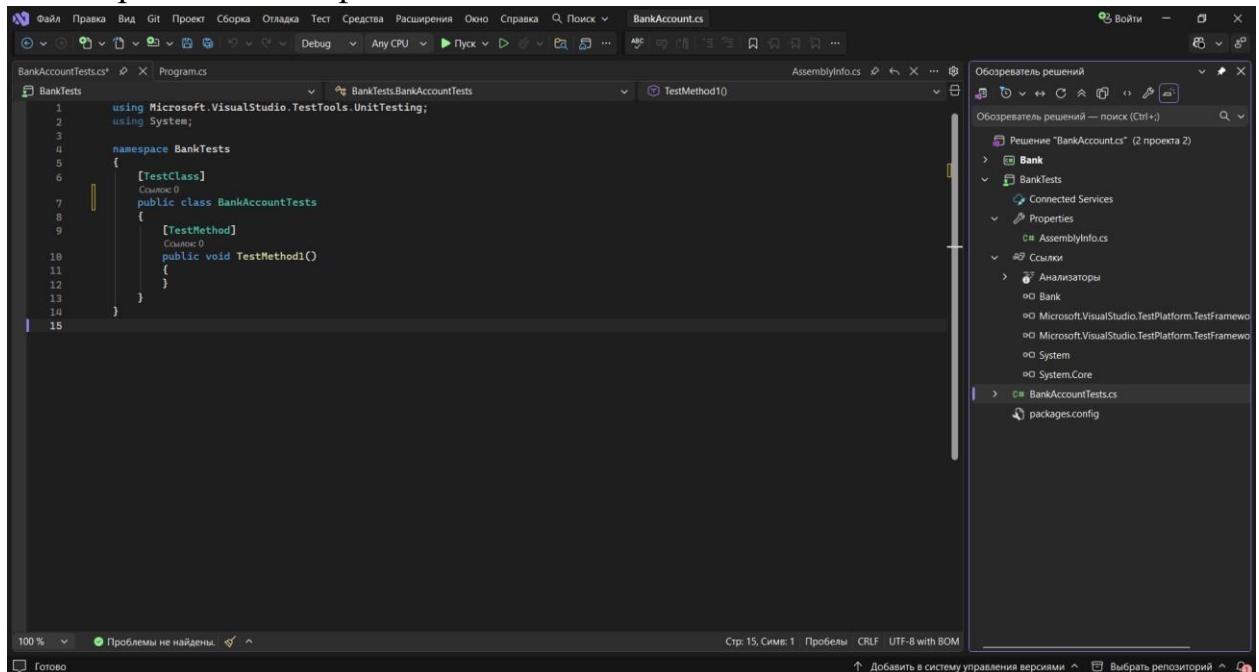


Рисунок 18

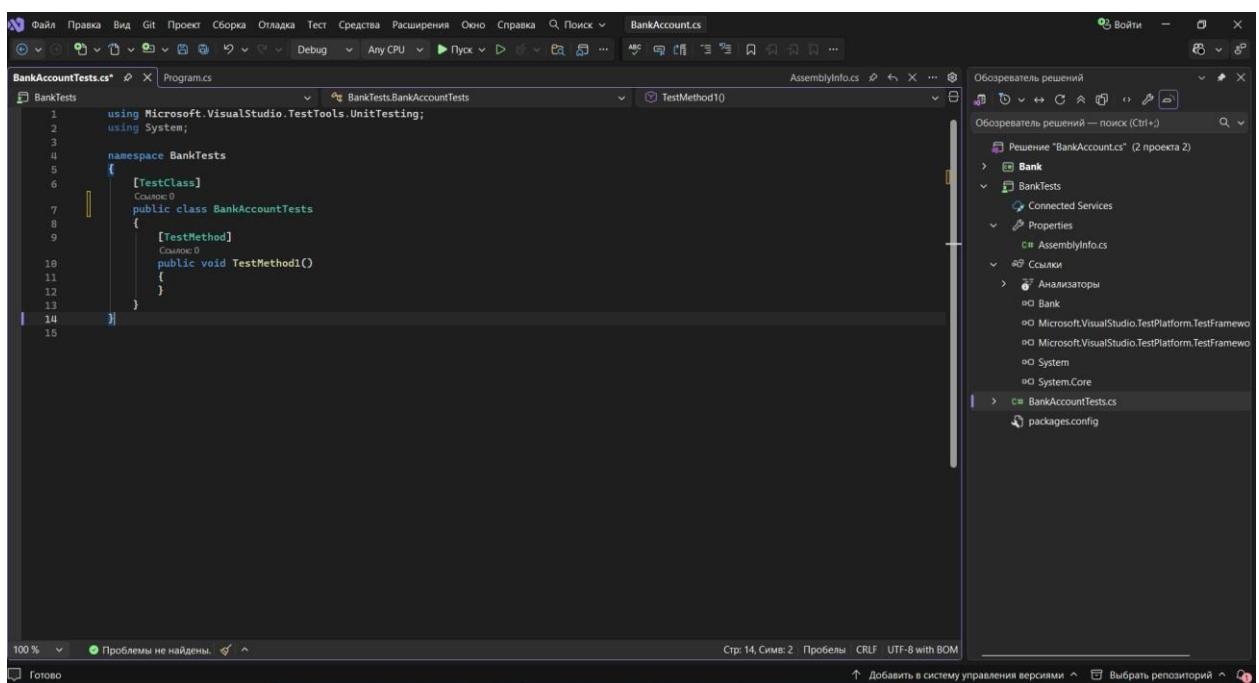
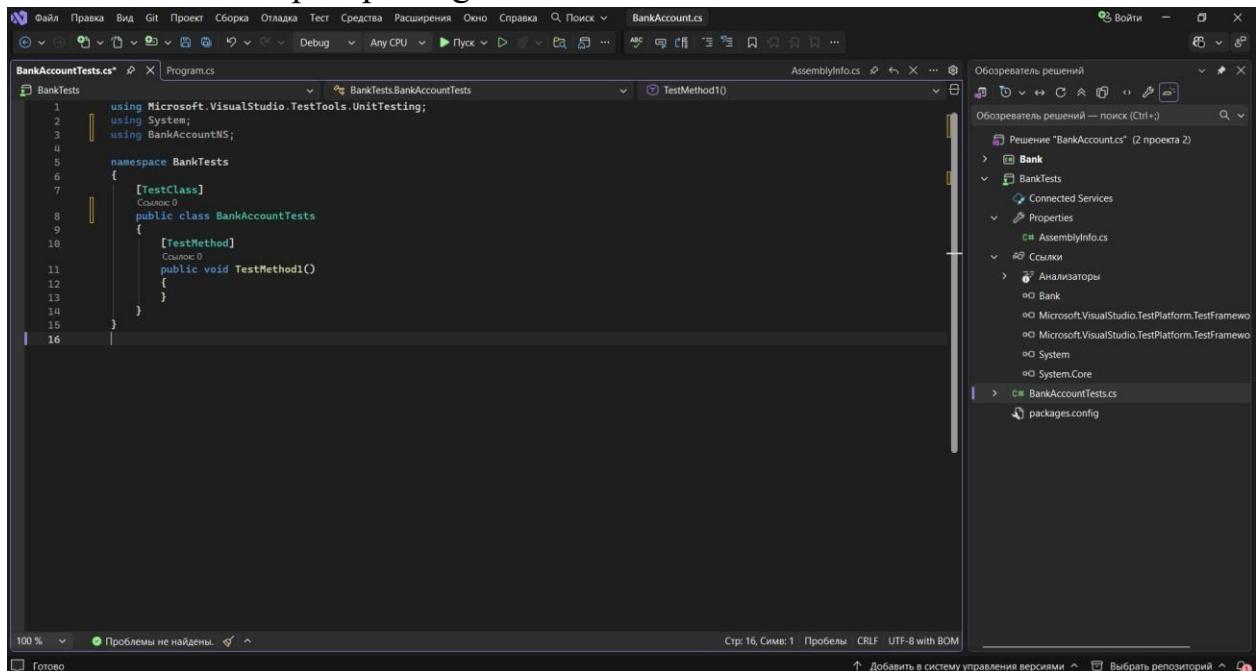


Рисунок 19

2. Я добавила оператор using.



The screenshot shows the Visual Studio IDE interface. The top menu bar includes: Файл, Правка, Вид, Git, Проект, Сборка, Отладка, Тест, Средства, Расширения, Окно, Справка, Поиск, and BankAccount.cs. The toolbar below has icons for Save, Undo, Redo, Cut, Copy, Paste, Find, Replace, and others. The main code editor window displays the file BankAccountTests.cs with the following content:

```
1  using Microsoft.VisualStudio.TestTools.UnitTesting;
2  using System;
3  using BankAccountNS;
4
5  namespace BankTests
6  {
7      [TestClass]
8      public class BankAccountTests
9      {
10          [TestMethod]
11          public void TestMethod1()
12          {
13          }
14      }
15  }
```

The status bar at the bottom indicates: 100%, Проблемы не найдены., Стр: 16, Симв: 1, Пробелы, CRLF, UTF-8 with BOM. To the right of the editor is the Solution Explorer window titled "Обозреватель решений". It shows a project named "Bank" with two files: AssemblyInfo.cs and BankAccountTests.cs. Under "Bank" are "Properties" and "Ссылки". Under "Ссылки" are "Анализаторы", "Bank", "Microsoft.VisualStudio.TestTools.UnitTesting", "System", and "System.Core". There is also a "packages.config" file listed under "Bank".

Рисунок 20

4. Создание метода теста.

1. Я ввела код:

```
using Microsoft.VisualStudio.TestTools.UnitTesting; using System; using BankAccountNS;

namespace BankTests
{
    [TestClass]
    public class BankAccountTests
    {
        [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");
        }
    }
}
```

The screenshot shows the Microsoft Visual Studio interface with the following details:

- Menu Bar:** Файл, Правка, Вид, Git, Проект, Сборка, Отладка, Тест, Средства, Расширения, Окно, Справка, Поиск.
- Toolbars:** Debug, Any CPU, Пуск, ...
- Code Editor:** BankAccountTests.cs (highlighted), Program.cs, AssemblyInfo.cs. The code in BankAccountTests.cs is as follows:

```
1  using Microsoft.VisualStudio.TestTools.UnitTesting;
2  using System;
3  using BankAccountNS;
4
5  namespace BankTests
6  {
7      [TestClass]
8      public class BankAccountTests
9      {
10         [TestMethod]
11         [DataSource()]
12         public void Debit_WithValidAmount_UpdatesBalance()
13         {
14             // Arrange
15             double beginningBalance = 11.99;
16             double debitAmount = 4.55;
17             double expected = 7.44;
18
19             BankAccount account = new BankAccount("Mr. Bryan Walton",
20             beginningBalance);
21
22             // Act
23             account.Debit(debitAmount);
24
25             // Assert
26             double actual = account.Balance;
27             Assert.AreEqual(expected, actual, 0.001, "Account not debited correctly");
28         }
29     }
30 }
```

- Solution Explorer:** Обзор решений (Solution Explorer) shows the solution structure:

 - Решение "BankAccountTests" (2 проекта)
 - Bank
 - BankTests
 - Connected Services
 - Properties
 - Ссылки
 - Анализаторы
 - Microsoft.VisualStudio.TestTools.UnitTesting
 - System
 - System.Core
 - BankAccountTests.cs
 - packages.config

- Status Bar:** Стр. 26 Симв. 3 Проблемы CRLF UTF-8 with BOM
- Bottom Bar:** Готово, Добавить в систему управления версиями, Выбрать репозиторий

Рисунок 21

5. Сборка и запуск теста.

1. Я использовала клавиши CTRL + SHIFT + B для того, чтобы произошла сборка (вывод был).

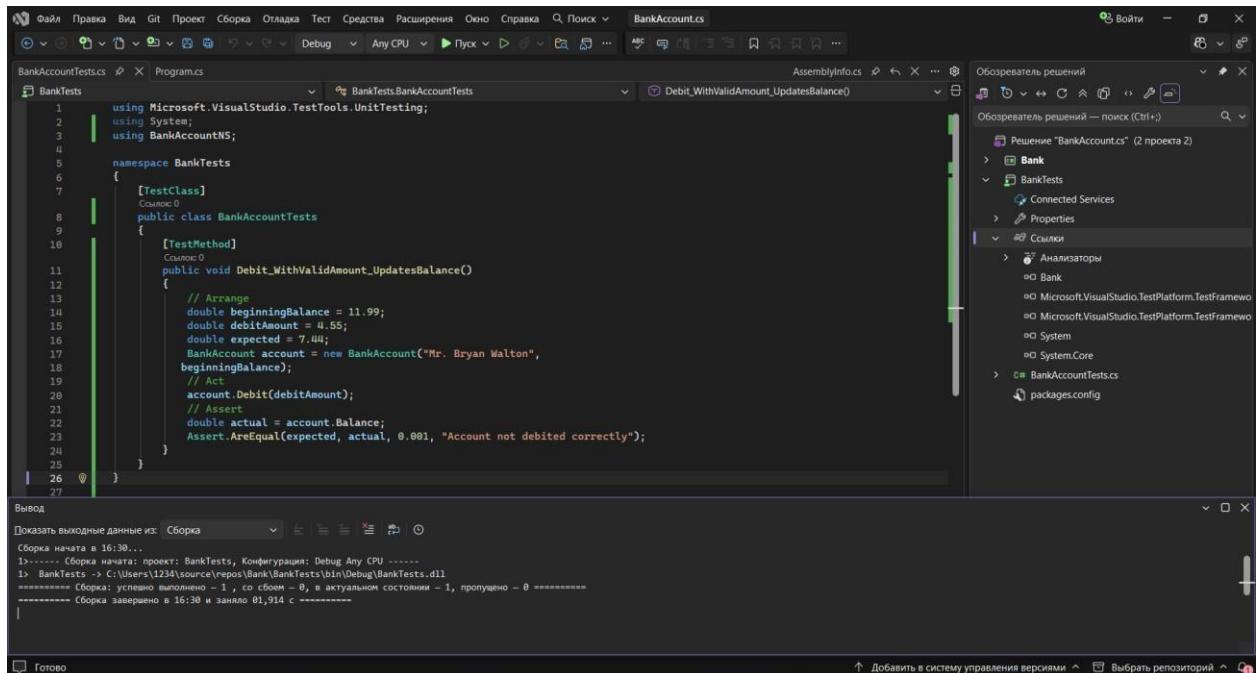


Рисунок 22

2. Я открыла Обозреватель тестов, выбрав Тест>Windows>Обозреватель тестов в верхней строке меню.

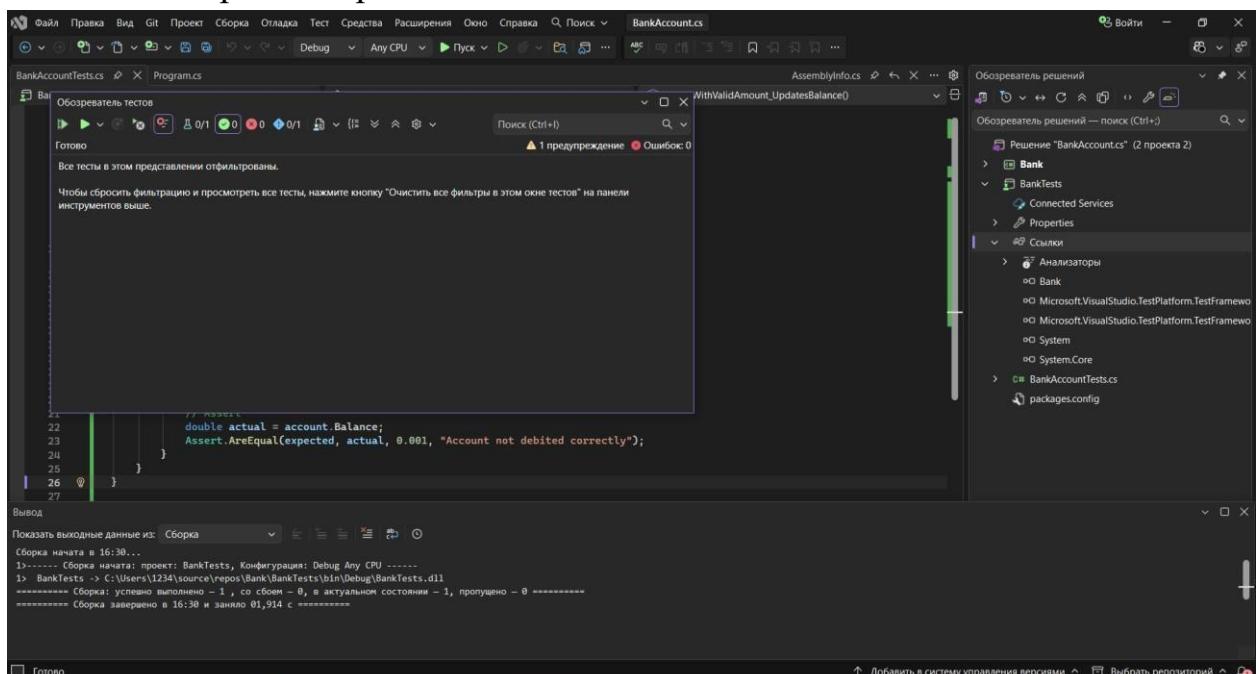


Рисунок 23

3. Я выбрала Запустить все, чтобы выполнить тест. В данном случае тест не пройден.

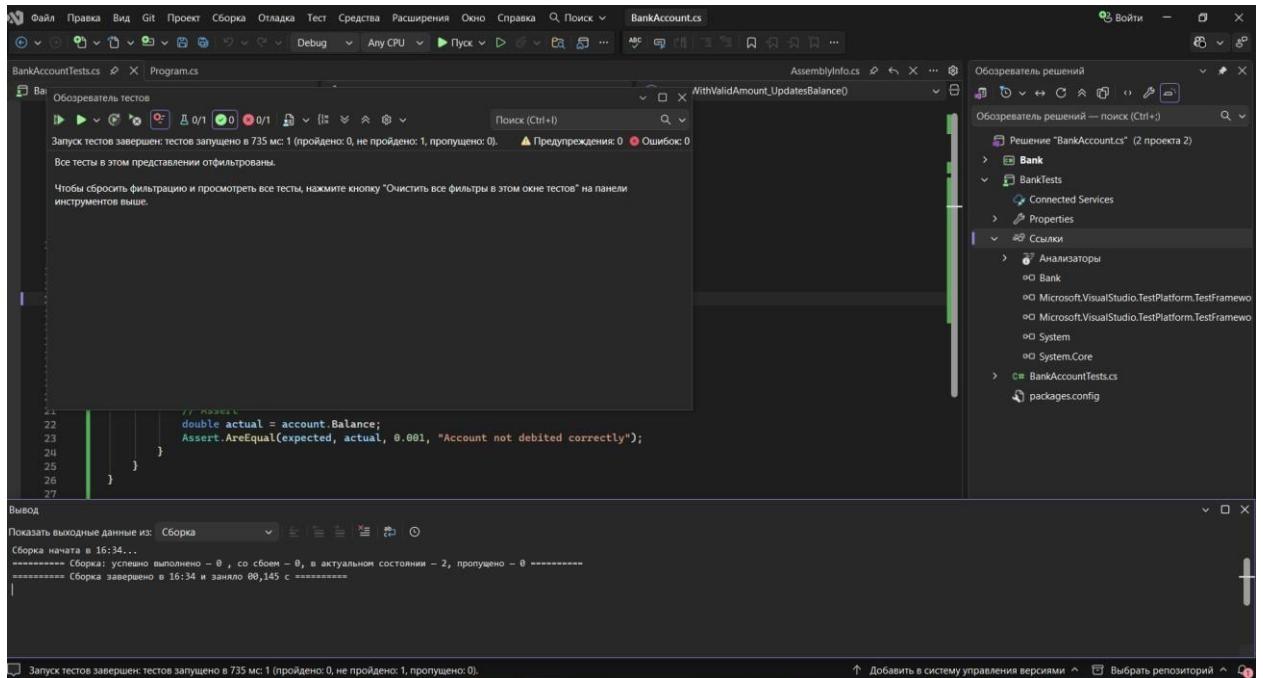


Рисунок 24

4. Я выбрала эту ошибку в обозревателе тестов для просмотра сведений в нижней части окна.

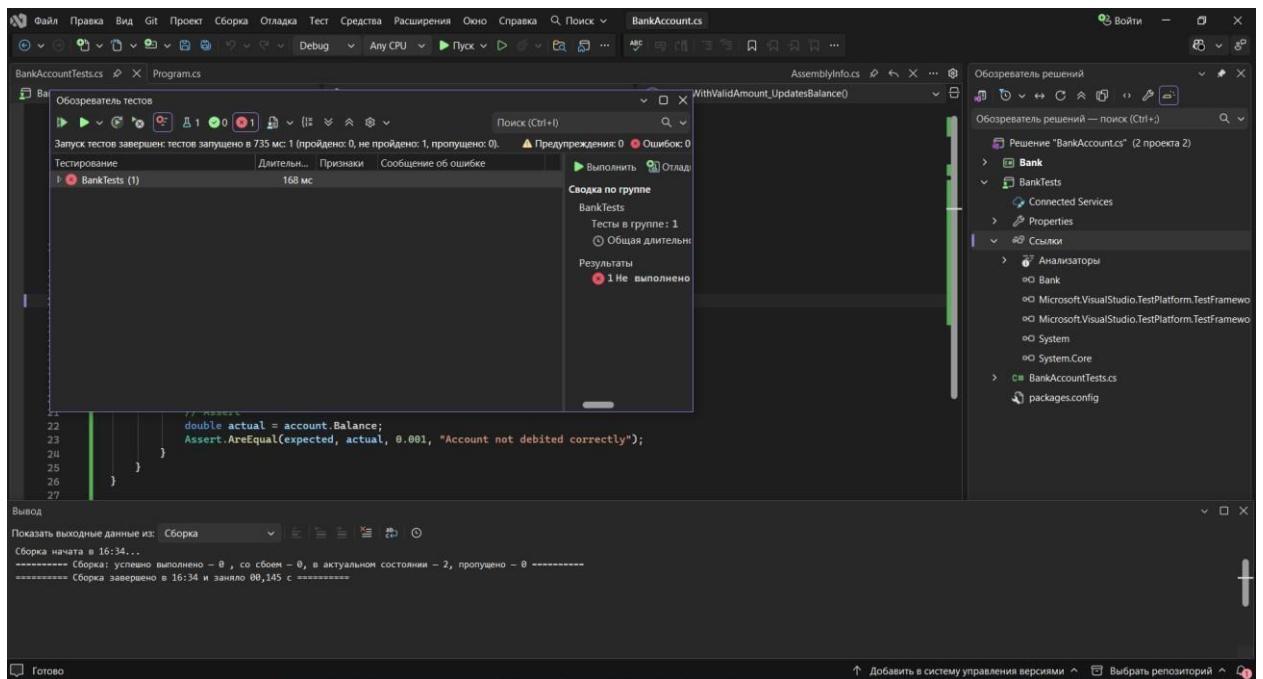


Рисунок 25

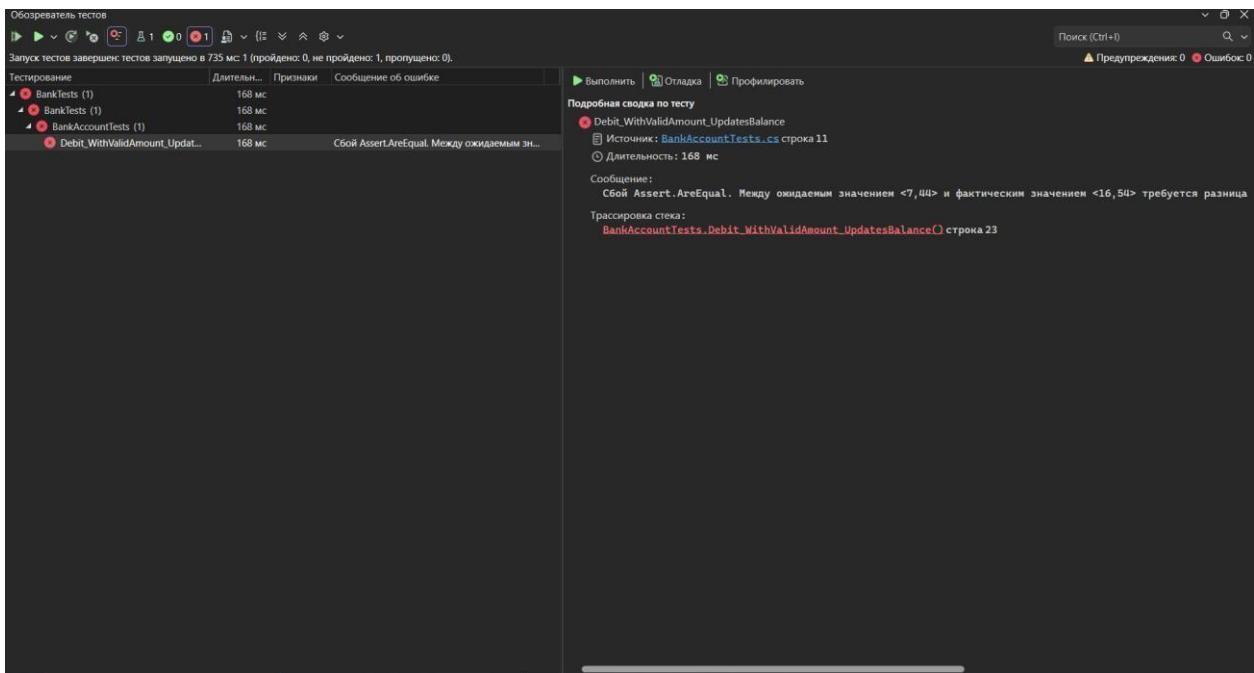


Рисунок 26

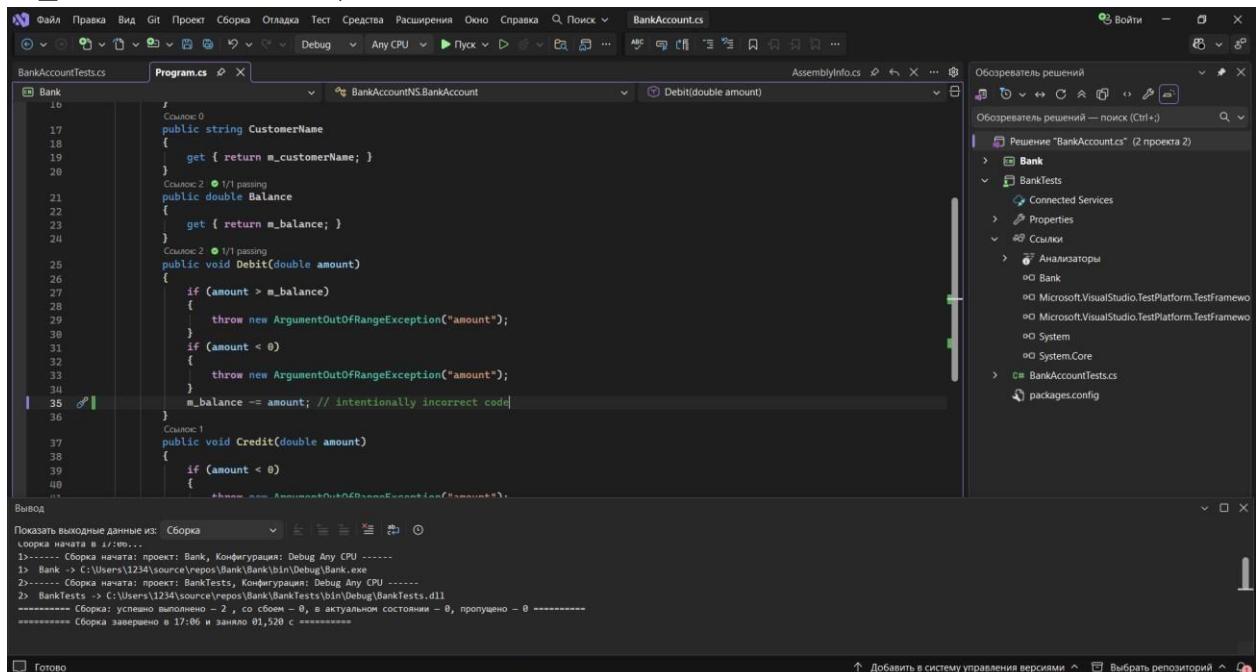
6. Исправление кода и повторный запуск тестов.

Я сделала исправление ошибки: В файле BankAccount.cs заменила строки с:

`m_balance += amount;`

на:

`m_balance -= amount;`

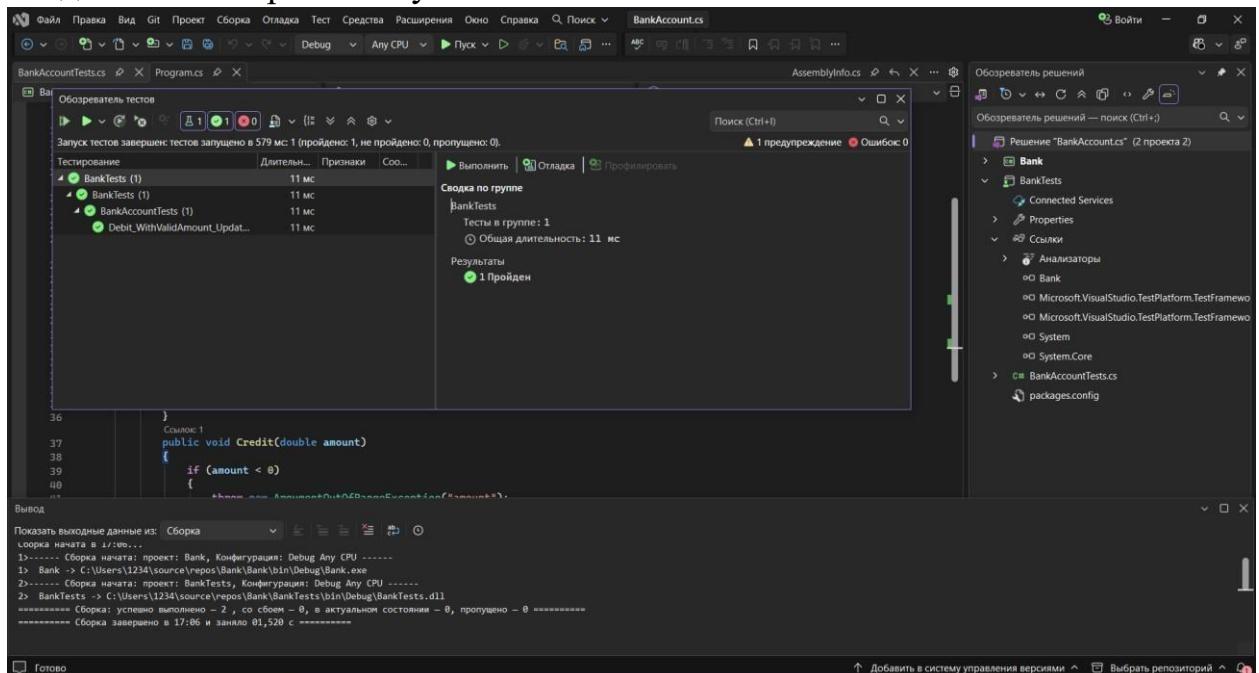


```
16     }
17     public string CustomerName
18     {
19         get { return _customerName; }
20     }
21     public double Balance
22     {
23         get { return _balance; }
24     }
25     public void Debit(double amount)
26     {
27         if (amount > _balance)
28         {
29             throw new ArgumentOutOfRangeException("amount");
30         }
31         if (amount < 0)
32         {
33             throw new ArgumentOutOfRangeException("amount");
34         }
35         _balance -= amount; // intentionally incorrect code
36     }
37     public void Credit(double amount)
38     {
39         if (amount < 0)
40         {
41             throw new ArgumentOutOfRangeException("amount");
42         }
43     }

```

Рисунок 27

Я сделала повторный запуск теста.



Запуск тестов завершен: тестов запущено в 579 мс 1 (пройдено: 1, не пройдено: 0, пропущено: 0).

Тестирование	Длительн...	Примеч...	Соо...
BankTests (1)	11 мс		
BankTests (1)	11 мс		
BankAccountTests (1)	11 мс		
Debit_WithValidAmount_UpdatingBalance	11 мс		

Сводка по группе

BankTests

Тесты в группе: 1

Общая длительность: 11 мс

Результаты

1 Пройден

Рисунок 28

7. Создание и запуск новых методов теста.

1. Я ввела код:

```
[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));
```

```
}
```

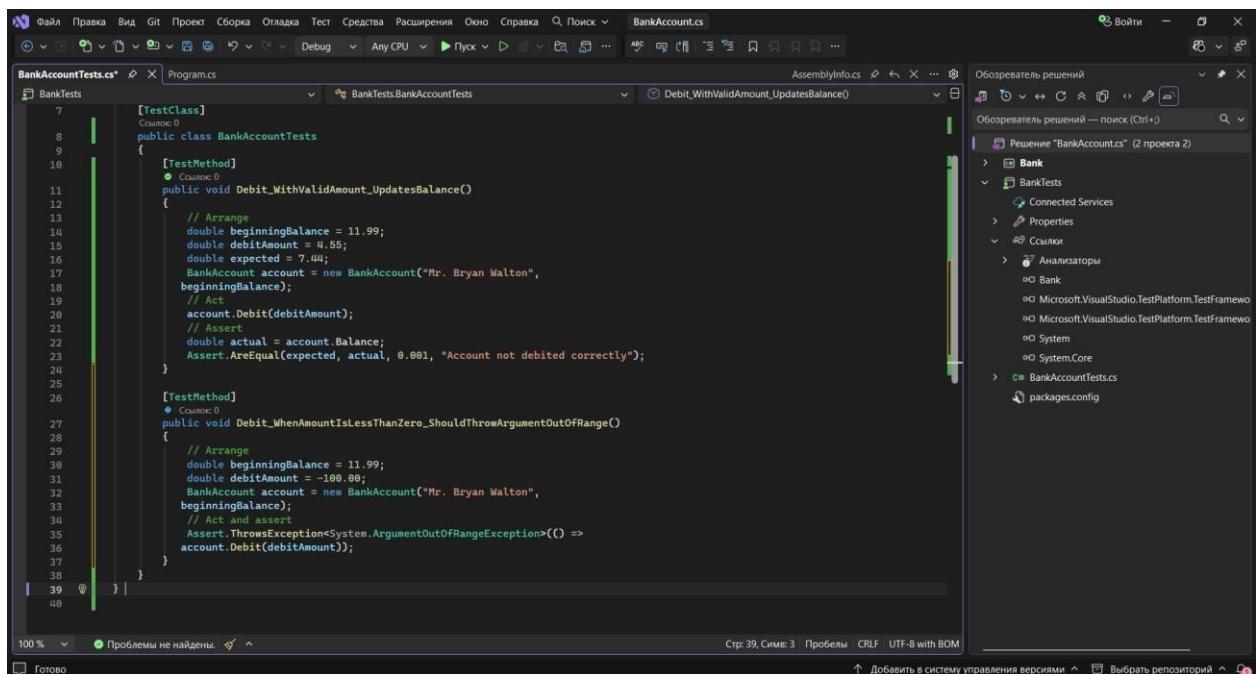


Рисунок 29

2. Выполнила проверку 2 тестов, убедилась, что они пройдены и нет ошибок.

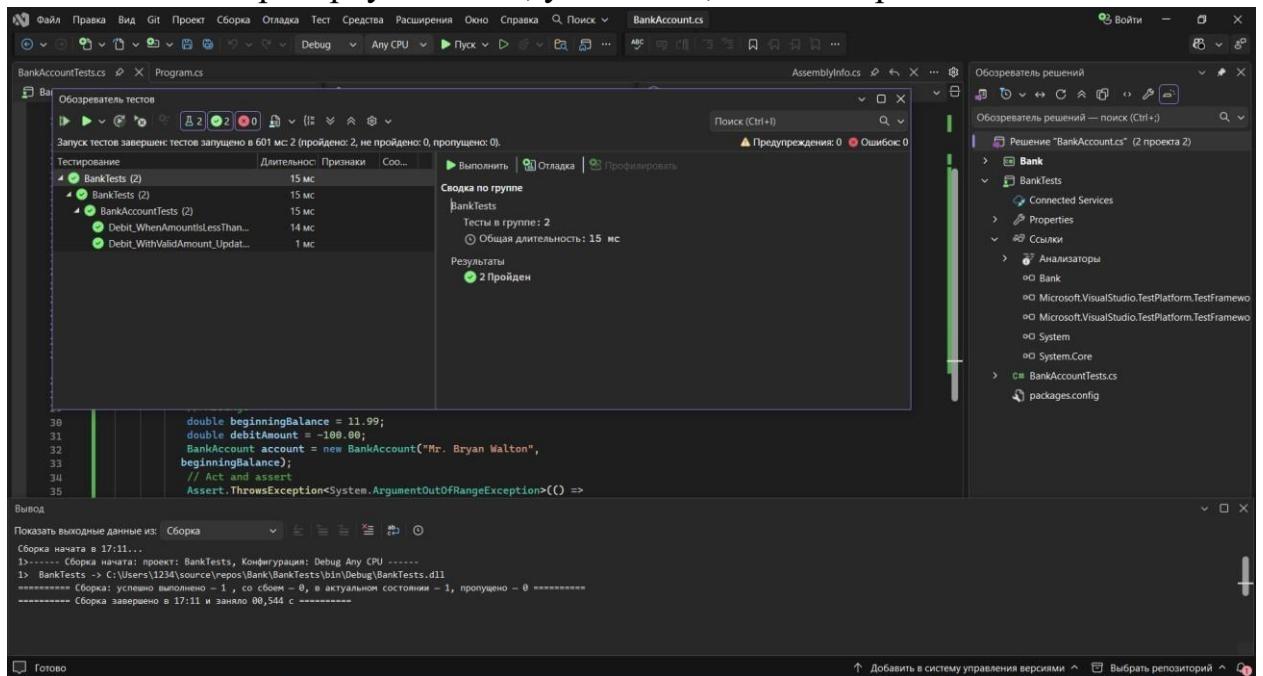


Рисунок 30

8. Рефакторинг тестируемого кода.

1. Я добавила это в тестируемый класс (BankAccount):

```
public const string DebitAmountExceedsBalanceMessage = "Debit amount exceeds balance";
```

```
public const string DebitAmountLessThanZeroMessage = "Debit amount is less than zero";
```

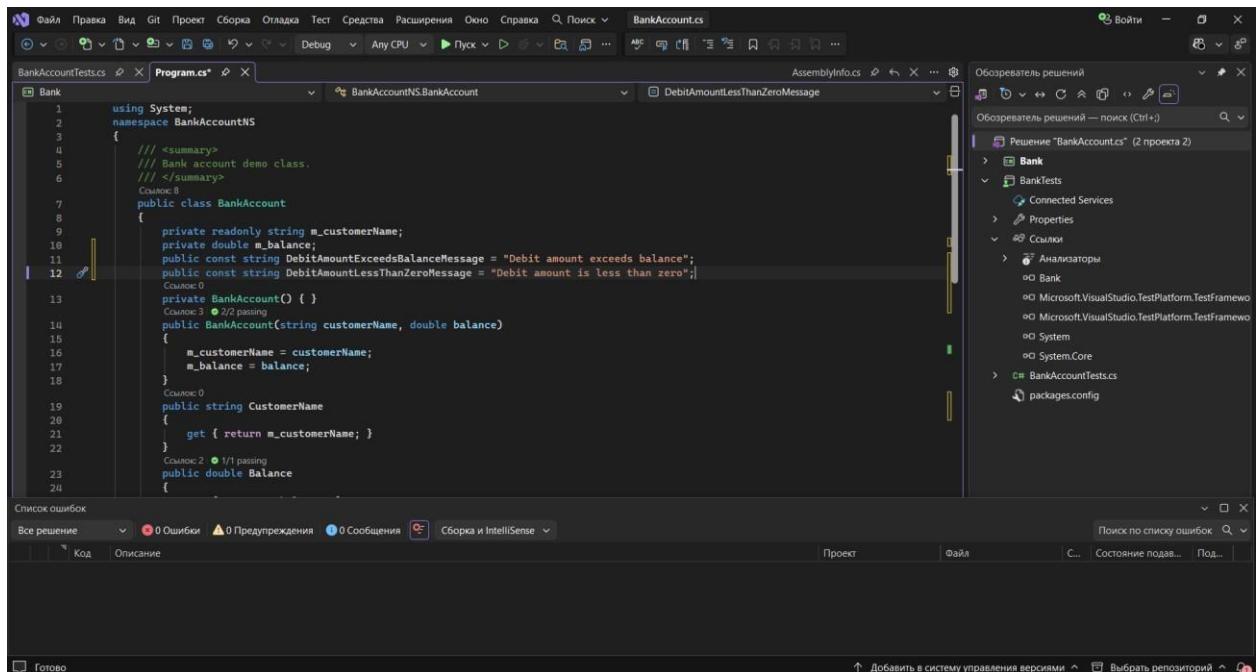


Рисунок 31

2. Я сделала замену двух условных операторов в методе Debit:

```
if (amount > m_balance)
{
    throw new System.ArgumentOutOfRangeException("amount", amount,
DebitAmountExceedsBalanceMessage);
}
if (amount < 0)
{
    throw new System.ArgumentOutOfRangeException("amount", amount,
DebitAmountLessThanZeroMessage);
}
```

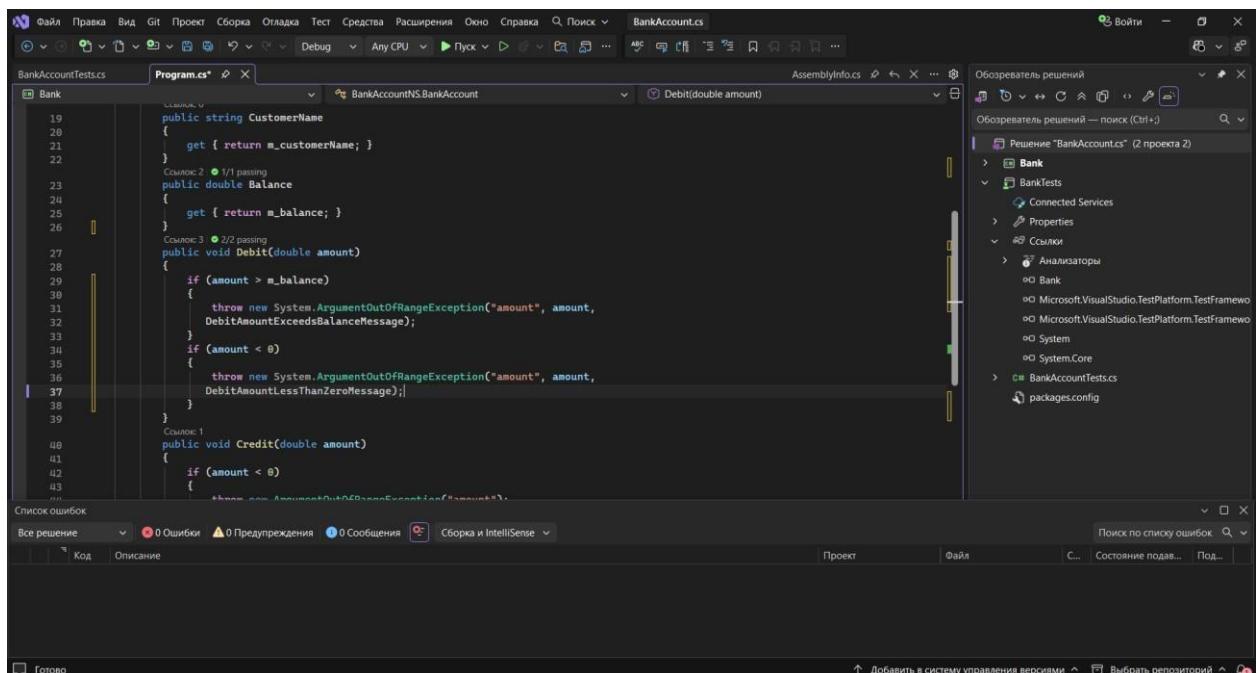


Рисунок 32

9. Рефакторинг тестовых методов.

Я ввела код:

```
[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);
    }
}
```

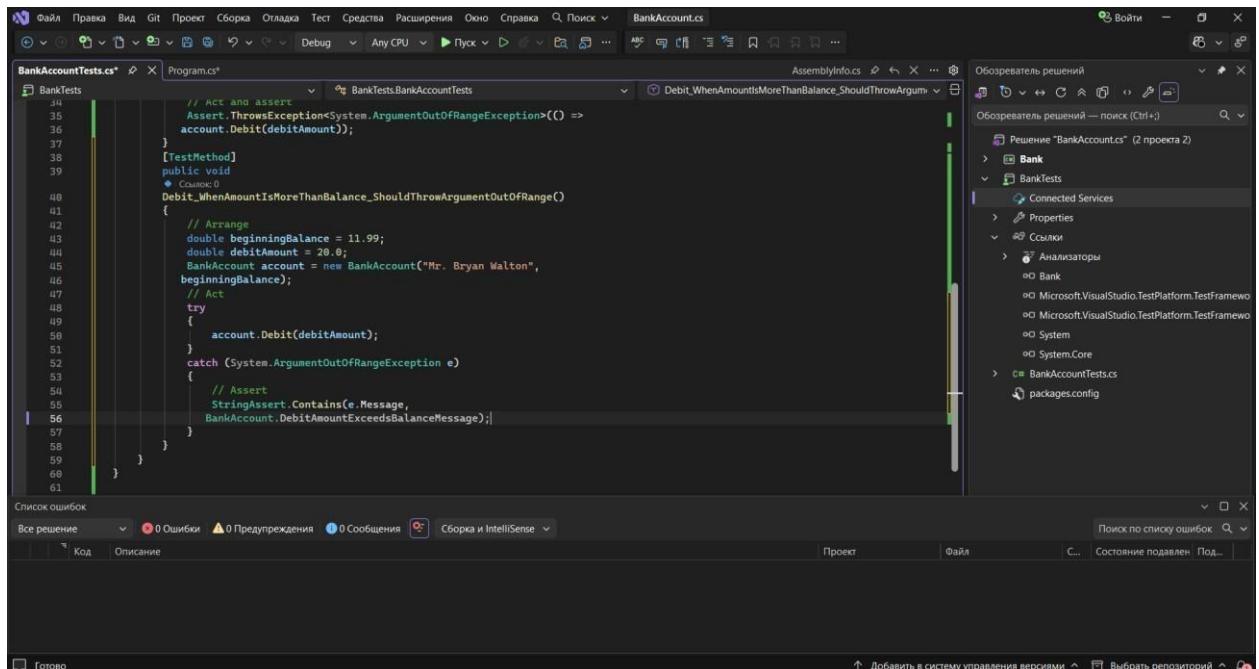


Рисунок 33

10. Повторное тестирование, переписывание и анализ.

1. Я редактировала код.

The screenshot shows the Microsoft Visual Studio interface. In the center, there is a code editor window displaying `BankAccountTests.cs`. The code contains a test method `Debit_WhenAmountIsMoreThanBalance_ShouldThrowArgumentOutOfRangeException()`. The code uses `Assert` statements to check if an exception is thrown when the debit amount exceeds the balance. The right side of the screen shows the **Solution Explorer** pane, which lists the project structure including files like `AssemblyInfo.cs`, `BankTests`, and `Bank`. At the bottom, there is a status bar with various icons and text.

Рисунок 34

2. Я провела проверку тестов. Я обнаружила ошибки в одном из тестов.

The screenshot shows the **Test Explorer** window in Microsoft Visual Studio. It displays a list of tests that have been run. One test, `Debit_WithValidAmount_UpdatesBalance()`, is shown as failed. A tooltip provides detailed information about the failure, stating: "Сбой Assert.AreEqual. Между ожидаемым значением <7,44> и фактическим значением <11,29> требуется разница". Below the tooltip, the stack trace indicates the error occurred at line 23 of `BankAccountTests.cs`. The bottom of the screen shows the Windows taskbar with various pinned icons.

Рисунок 35

3. Я приняла решение анализировать ошибку и искать причины её возникновения, и пыталась решить возникшие ошибки.

Я добавила строку в код для устранения ошибки: `m_balance -= amount;`

4. Я совершила запуск всех тестов (повторная проверка).

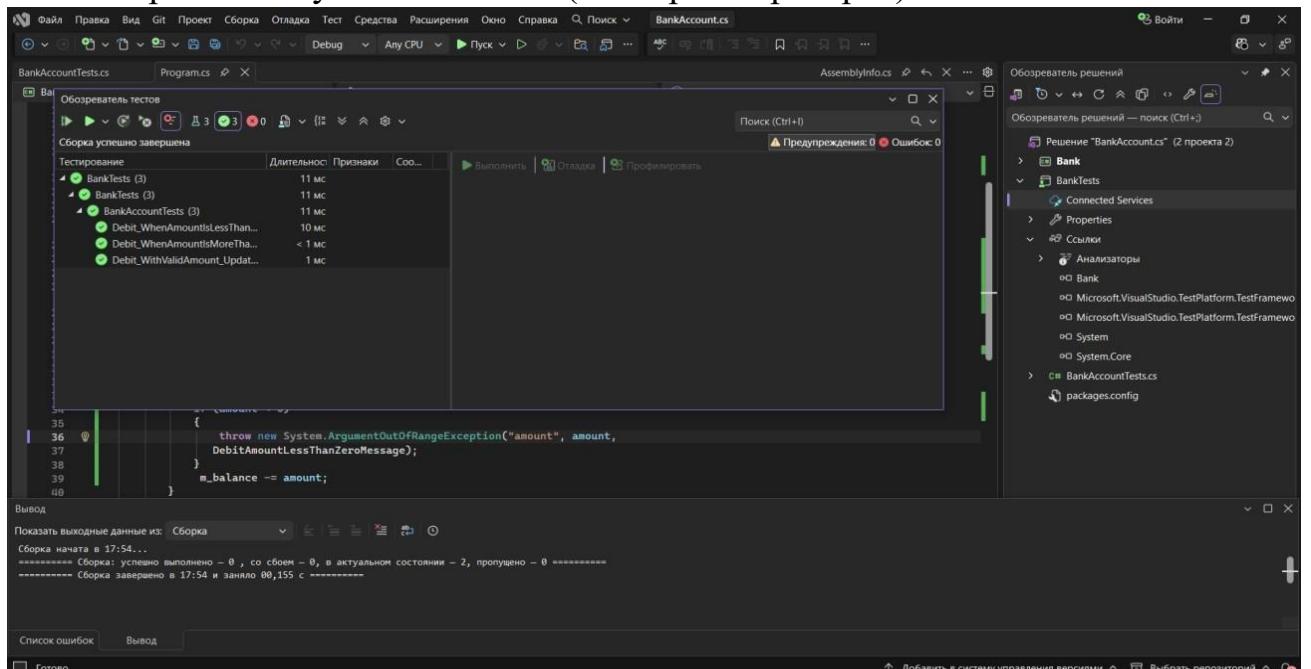


Рисунок 36

5. Я провела анализирование вывода, ошибка устранена. Все тесты выполнены правильно.

6. Проверка работы программы.

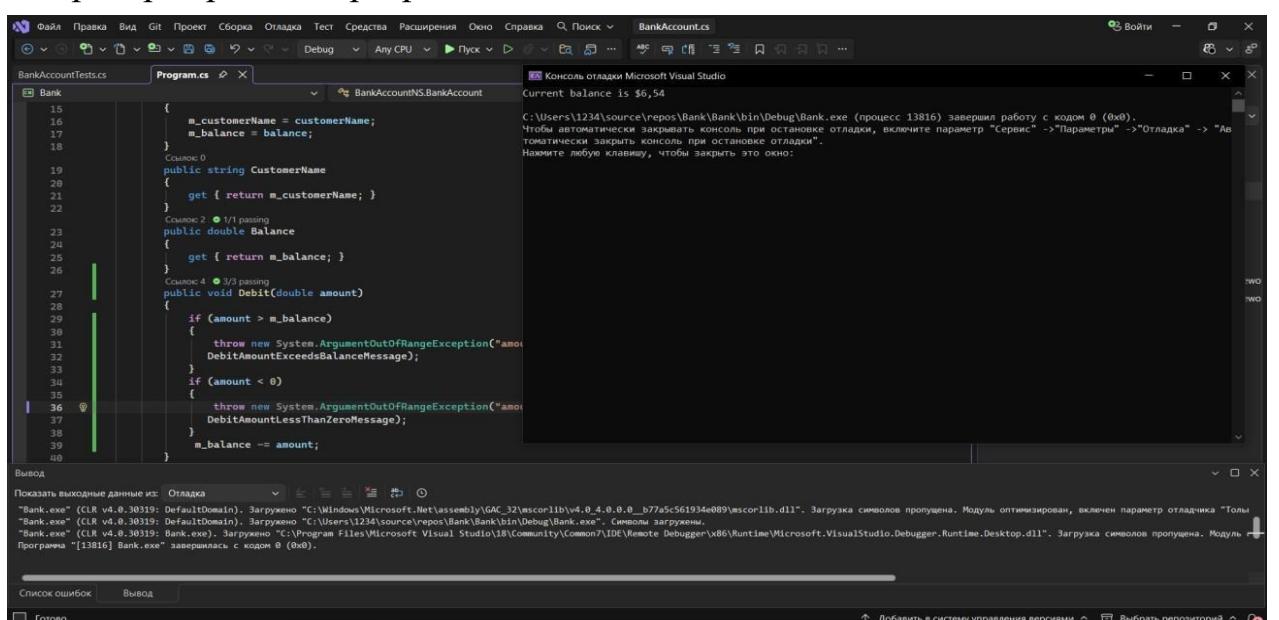


Рисунок 37

Заключение.

Выполняя практическую работу №1 по теме «Средства тестирования Visual Studio-2022», я освоила работу со средой тестирования. Научилась создавать и запускать модульные тесты, и выполнять рефакторинг. Таким образом, я смогла усовершенствовать тестовый код, что привело к созданию более надёжного информативного метода теста. Но самое главное, что в результате был улучшен тестируемый код. Я научилась составлять официальную документацию. Осознала, что всегда нужно перепроверять свою работу множество раз и что заниматься тестированием очень классно.

