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)

Сопроводительная записка по дисциплине

«Поддержка и тестирование программных модулей»

по теме:

«Создание и запуск модульных тестов для управляемого кода»

Руководитель

_____/Берестов Д.Б./

«___» _____2026

Исполнитель

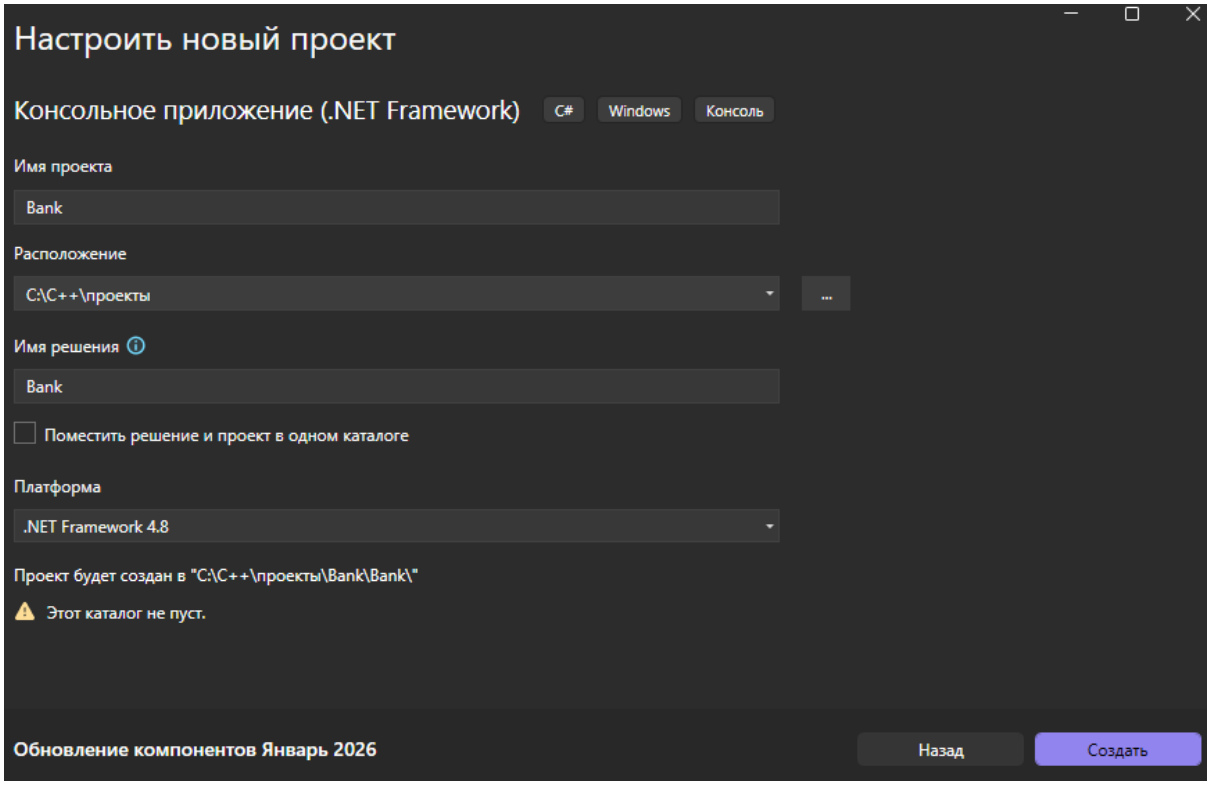
_____/Муратов Л.А./

«___» _____2026 г.

Пермь, 2026 г.

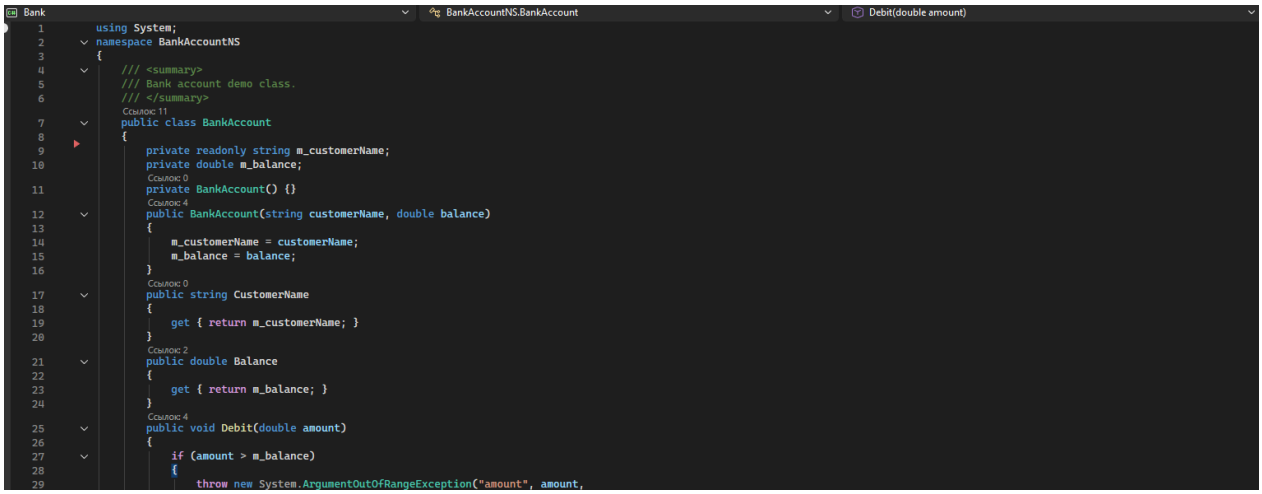
Создание проекта для тестирования

Создаю проект (см. Рис. 1)



(Рис. 1)

Заменяю содержимое файла Program.cs кодом, предложенным методичкой. (см. Рис. 2)



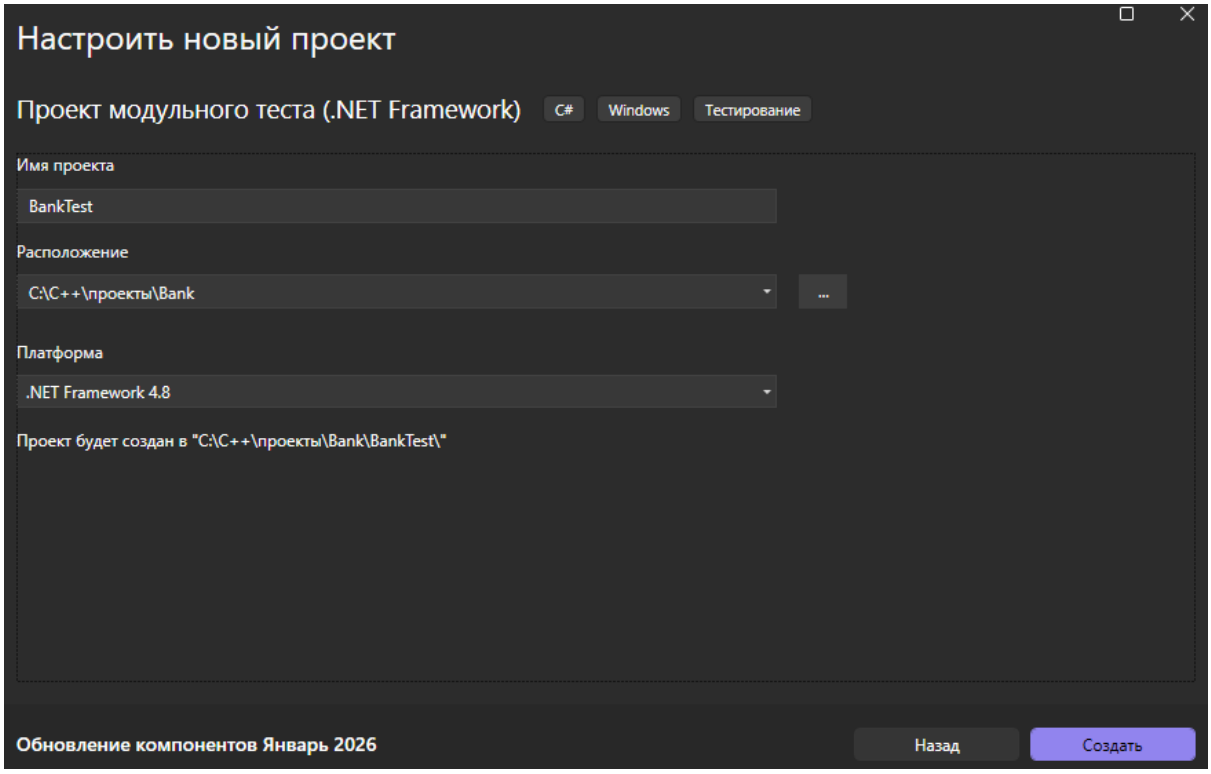
(Рис. 2)

Переименовываю файл Programm.cs в BankAccount.cs в обозревателе решений.

В меню сборка нажимаю построить решение.

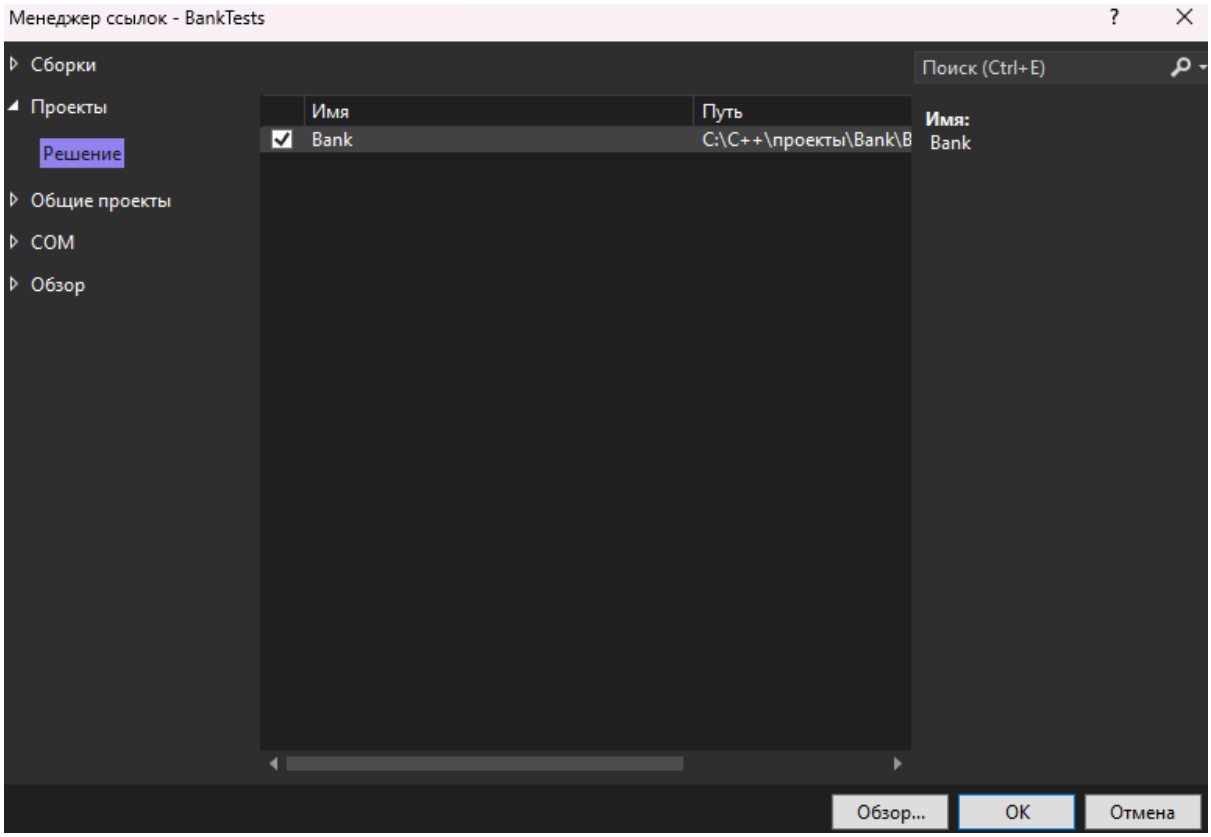
Создание проекта модульного теста

Создаю проект модульного теста (см. Рис. 3)



(Рис. 3)

В проекте BankTests добавляю ссылку на проект Банк. (см. Рис. 4)

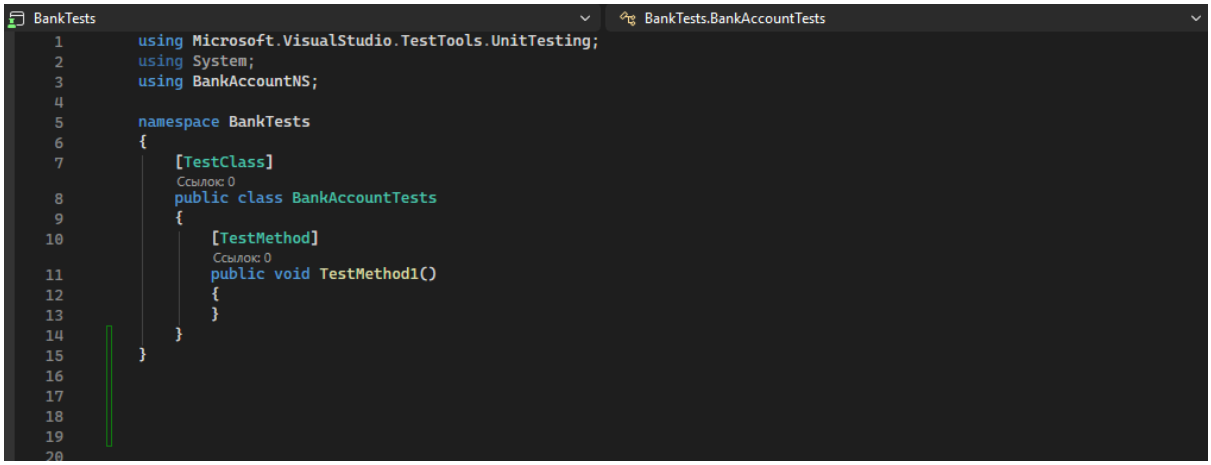


(Рис. 4)

Создание тестового класса

Переименовываю файл UnitTest1.cs в BankAccountTests.cs.

Файл BankAccountTests.cs теперь содержит следующий код (см. Рис. 5).



(Рис. 5)

Добавляю оператор using BankAccountNS; (см. Рис. 6)

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

(Рис. 6)

Создание метода теста

Добавляю предложенный из методички метод в класс BankAccountTest. (см. Рис. 7)

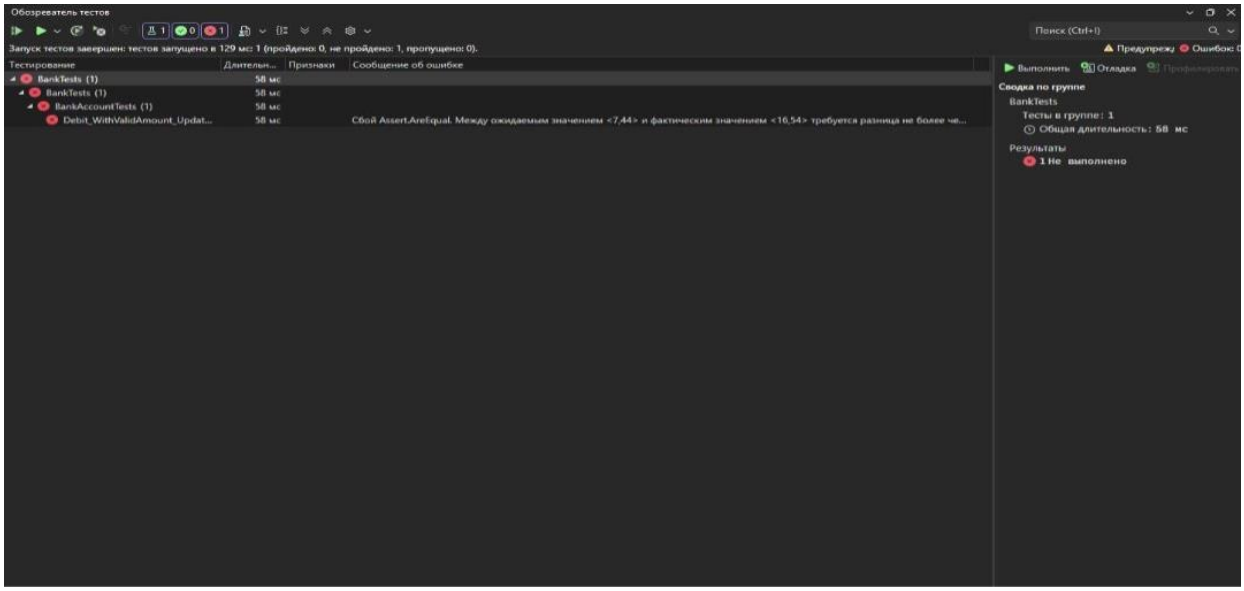
```
[TestMethod]
Ссылок: 0
public void Debit_WithValidAmount_UpdatesBalance()
{
    // Arrange
    double beginningBalance = 11.99;
    double debitAmount = 4.55;
    double expected = 7.44;
    BankAccount account = new BankAccount("Mr. Bryan Walton",
   beginningBalance);
    // Act
    account.Debit(debitAmount);
    // Assert
    double actual = account.Balance;
    Assert.AreEqual(expected, actual, 0.001, "Account not debited correctly");
}
```

(Рис. 7)

Сборка и запуск теста

В меню *Сборка* нажмимаю *Построить решение*.

Выбераю *Запустить все*, чтобы выполнить тест. (см. Рис. 8)



(Рис. 8)

Исправляю ошибки.

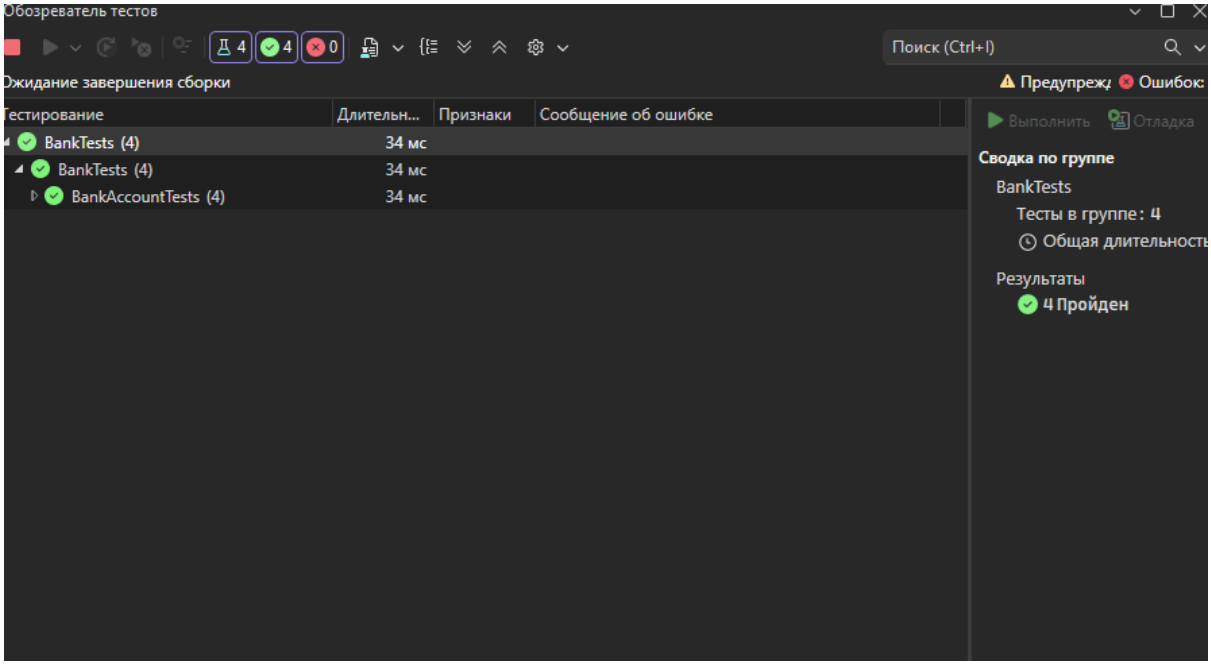
Чтобы исправить эту ошибку, в файле BankAccount.cs замените строку:

m_balance += amount; на *m_balance -= amount;* (см. Рис. 9)

```
    m_balance -= amount; // intentionally incorrect code
}
Ссылок: 1
public void Credit(double amount)
{
    if (amount < 0)
    {
        throw new ArgumentOutOfRangeException("amount");
    }
    m_balance -= amount;
}
```

(Рис. 9)

Повторно запускаю тест. (см. Рис. 10)



(Рис. 10)

Создание и запуск новых методов теста

Создаю метод теста для проверки правильного поведения в случае, когда сумма по дебету меньше нуля (см. Рис. 11)

```
[TestMethod]
Ссылок: 0
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));
}
```

(Рис. 11)

Создадим метод теста для проверки правильного поведения в случае, когда размер списания превышает баланс: (см. Рис. 12)

```
        [TestMethod]
        Ссылок: 0
        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("The expected exception was not thrown.");
        }
    }
}
```

(Рис. 12)

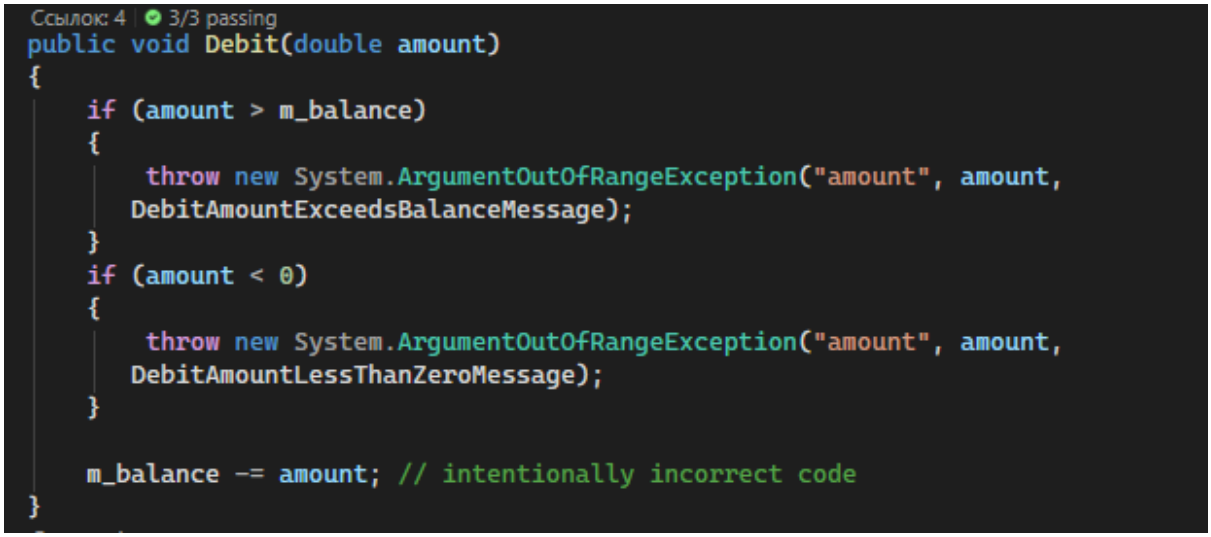
Рефакторинг тестируемого кода

Сначала определяю две константы для сообщений об ошибках в области видимости класса. Добавляю это в тестируемый класс (BankAccount): (см. Рис. 13)

```
using System;
namespace BankAccountNS
{
    /// <summary>
    /// Bank account demo class.
    /// </summary>
    Ссылок: 11
    public class BankAccount
    {
        public const string DebitAmountExceedsBalanceMessage = "Debit amount exceeds balance";
        public const string DebitAmountLessThanZeroMessage = "Debit amount is less than zero";
        private readonly string m_customerName;
        private double m_balance;
        Ссылок: 0
        private BankAccount() {}
        Ссылок: 4  3/3 passing
        public BankAccount(string customerName, double balance)
        {
```

(Рис. 13)

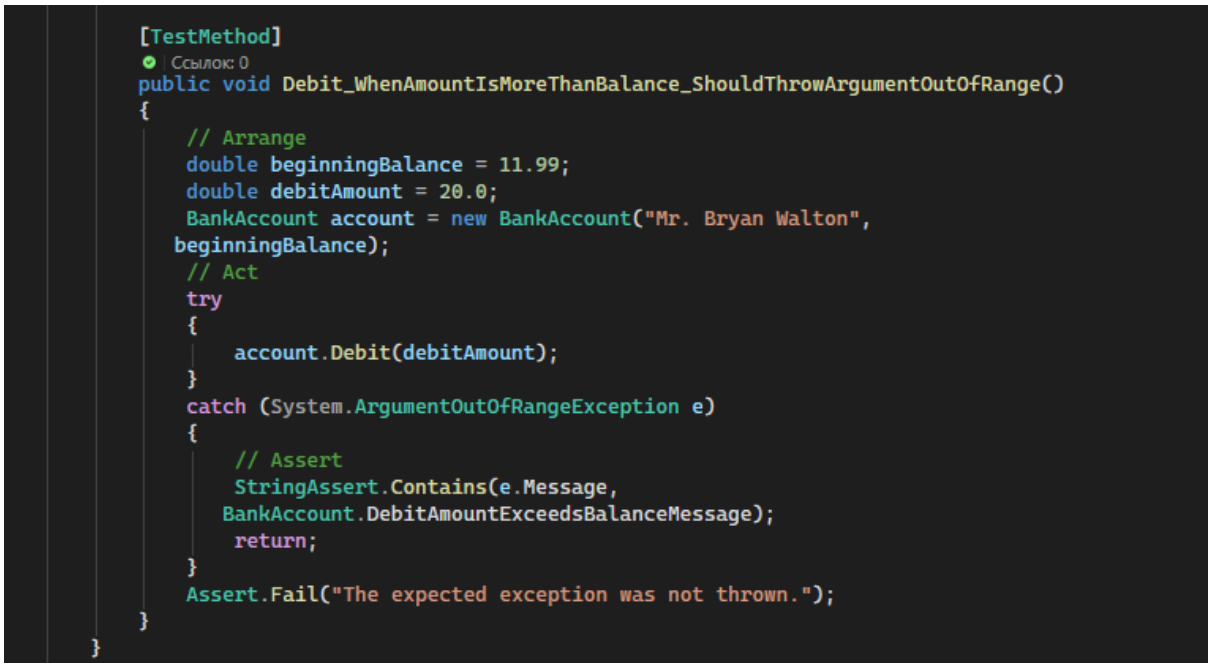
Затем изменяю два условных оператора в методе Debit: (см. Рис. 14)



(Рис. 14)

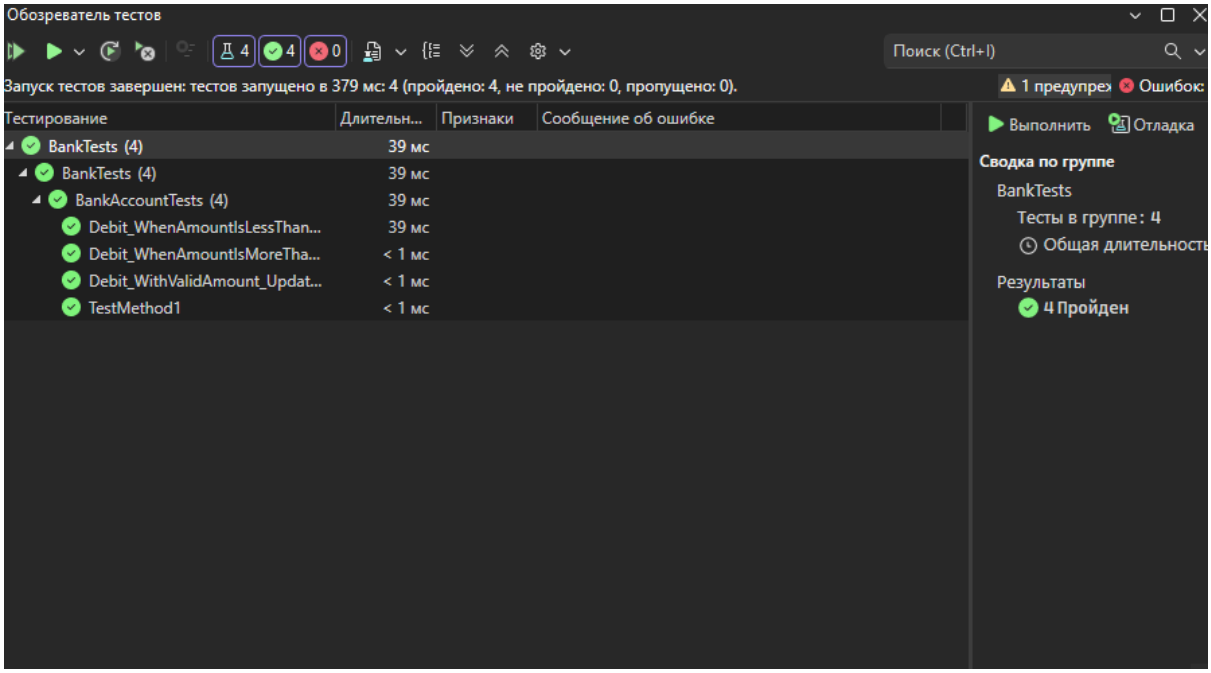
Рефакторинг тестовых методов

Выполняю рефакторинг методов теста, удалив вызов Assert.ThrowsException. Заключаю вызов Debit() в блок try/catch , перехватите конкретное ожидаемое исключение и проверьте соответствующее ему сообщение. (см. Рис. 15)



(Рис. 15)

Провожу тестирование, тестирование выполнено. (см. Рис. 16)


Обозреватель тестов
4
4
0
Поиск (Ctrl+I)
Запуск тестов завершен: тестов запущено в 379 мс: 4 (пройдено: 4, не пройдено: 0, пропущено: 0).
1 предупре
Ошибок:
Тестирование
Длительн...
Признаки
Сообщение об ошибке
BankTests (4) 39 мс
BankTests (4) 39 мс
BankAccountTests (4) 39 мс
Debit_WhenAmountIsLessThan... 39 мс
Debit_WhenAmountIsMoreTha... < 1 мс
Debit_WithValidAmount_Updat... < 1 мс
TestMethod1 < 1 мс
Выполнить
Отладка
Сводка по группе
BankTests
Тесты в группе: 4
Общая длительность
Результаты
4 Пройден

(Рис. 16)