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»

Руководитель: Д.Б. Берестов /И.О. Фамилия/
“__26__”_февраля_2026г

Исполнитель: А.А. Визгалова /И.О. Фамилия/
“__26__” _февраля_ 2026 г.

Пермь 2026 г.

Оглавление

Целью практической работы является освоение процесса создания, запуска и настройки модульных тестов с использованием платформы модульного тестирования Майкрософт для управляемого кода (MSTest) и обозревателя тестов Visual Studio. В ходе работы выполняется проверка кода проекта, находящегося в стадии разработки, путем написания тестов, их выполнения и анализа полученных результатов.

Задание

Разработать модульные тесты для проверки поведения метода Debit класса BankAccount. В соответствии с логикой работы метода, тесты должны проверять три ключевых сценария:

1. Корректность выполнения операции: при допустимой сумме списания (меньше баланса и больше нуля) остаток на счете уменьшается на соответствующую величину.

2. Обработка исключения при отрицательной сумме: при попытке списания суммы меньше нуля метод должен создавать исключение ArgumentOutOfRangeException.

3. Обработка исключения при превышении баланса: при попытке списания суммы, превышающей текущий баланс, метод должен создавать исключение ArgumentOutOfRangeException.

Структура проекта

Bank – основной проект (библиотека классов), содержащий тестируемый класс BankAccount с методом Debit.

BankTests – проект модульного теста, содержащий тестовый класс BankAccountTests с методами для проверки функциональности класса BankAccount.

Такая структура, при которой тестовый проект создается в том же решении и содержит ссылку на тестируемый проект, позволяет изолированно проверить логику работы кода, что является основным принципом модульного тестирования.

Создание проекта в Visual Studio

Шаг 1. Открытие Visual Studio

Нужно открыть Visual Studio перед нами откроется главный экран на котором мы должны создать новый проект. (Смотреть Рис.1)

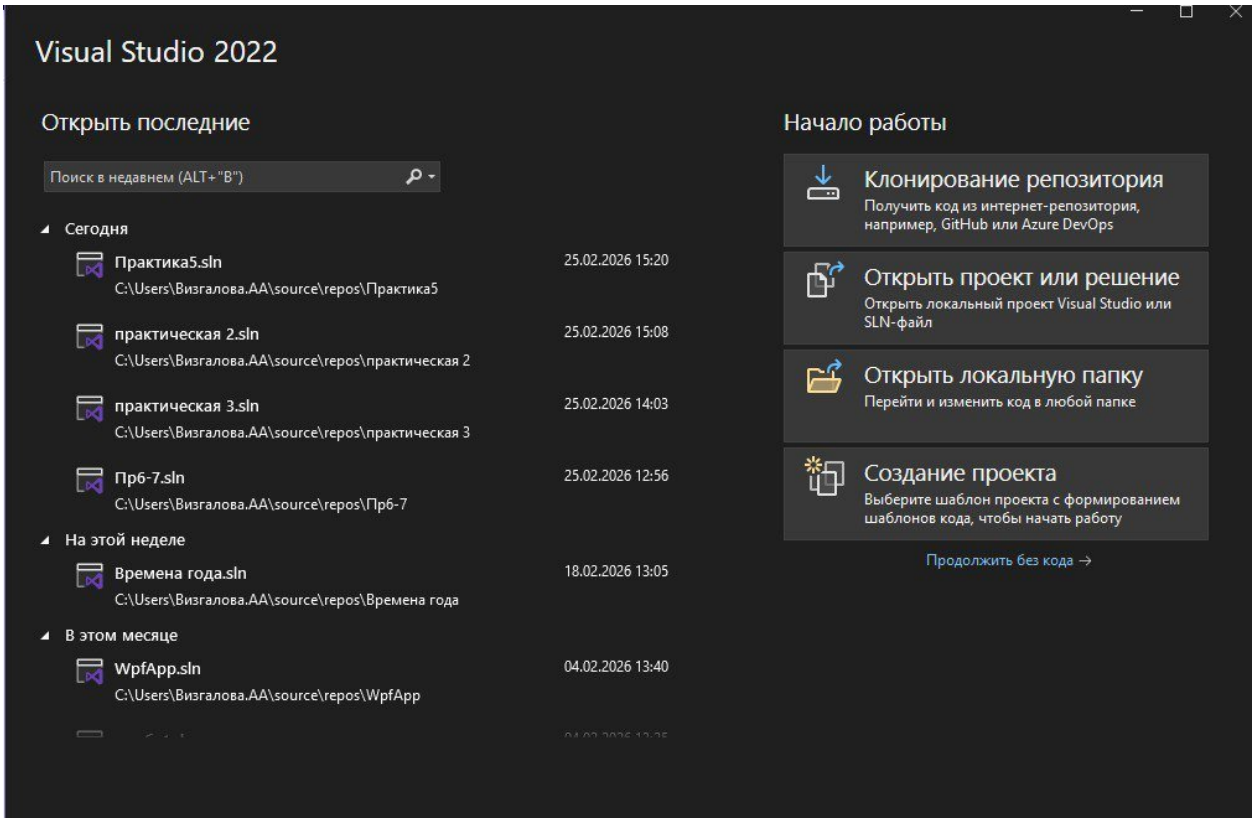

Рис.1 - Открытие Visual Studio

Дальше выбрать Консольное приложение, назвав его Bank. (Смотреть Рис.2)

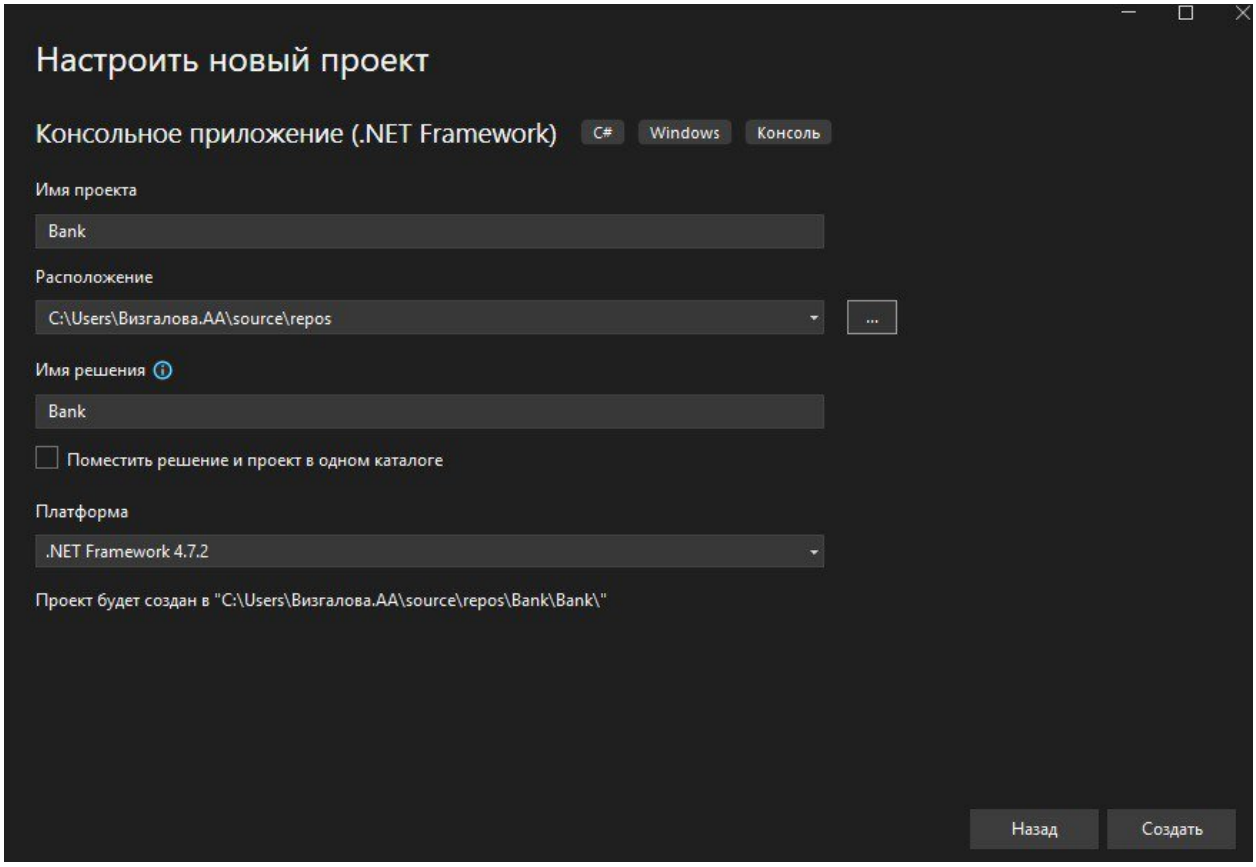

Рис.2 – Настраивание проекта

Добавляем модульные тесты в проект (Смотреть Рис.3)

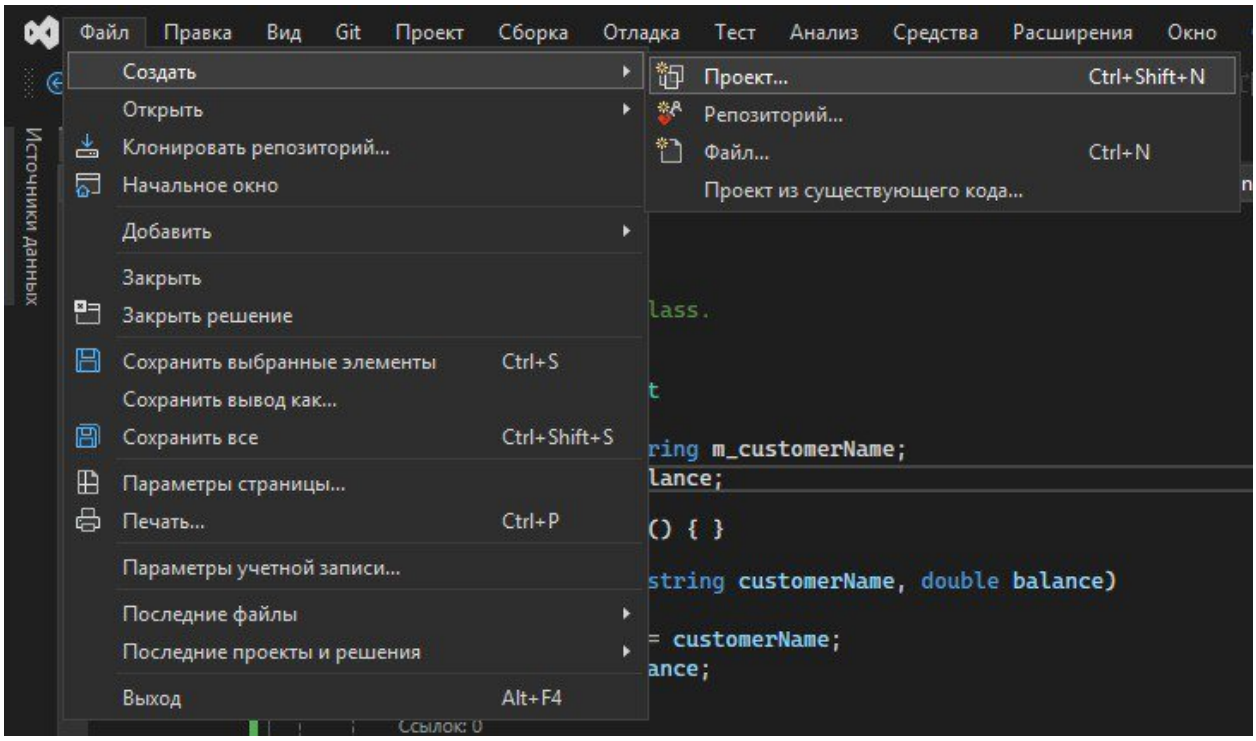

Рис.3 - Добавление модульных тестов в проект

Шаг 2: Создание проекта модульного теста

1. В меню Файл выбираем Добавить - Создать проект.

2. Находим и выбираем шаблон Проект модульного теста MSTest (.NET Core). Затем называем его BankTests.

3. В проекте BankTests добавляем ссылку на проект Bank.

Правой Кнопкой Мыши по проекту по BankTests - Добавить ссылку - Проекты - отмечаем Bank. (Смотреть Рис.4)

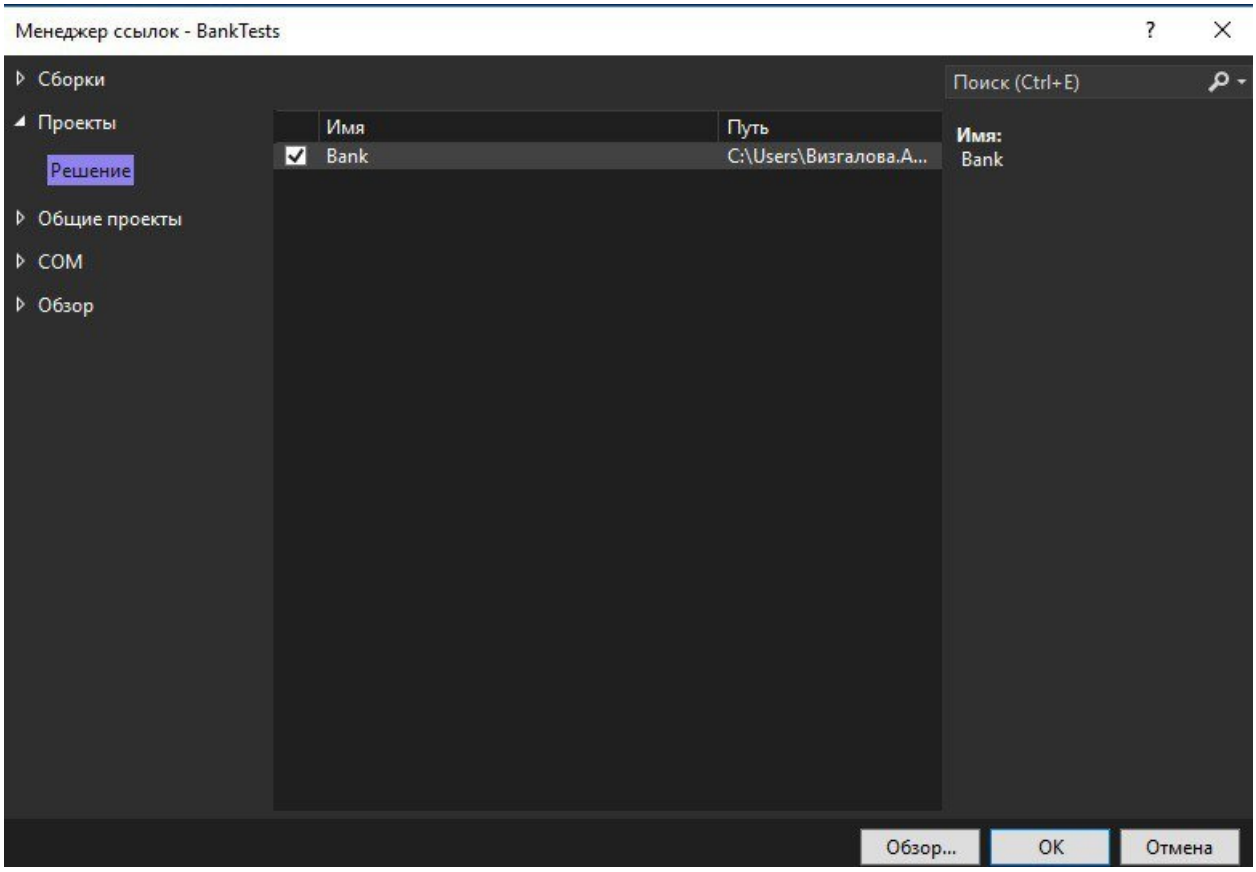

Рис.4 – Добавление ссылки

.

Шаг 3: Первый запуск тестов

После сборки решения (Сборка - Пересобрать решение) был открыт Обозреватель тестов (Тест - Обозреватель тестов) и выполнен запуск всех тестов (Смотреть Рис.5).

На рис.5 представлен результат первого запуска: ошибка. 1 тест не пройден - это соответствует ожидаемому поведению, описанному в методичке, так как в исходной реализации метода Debit была намеренно допущена ошибка (баланс увеличивался вместо уменьшения).

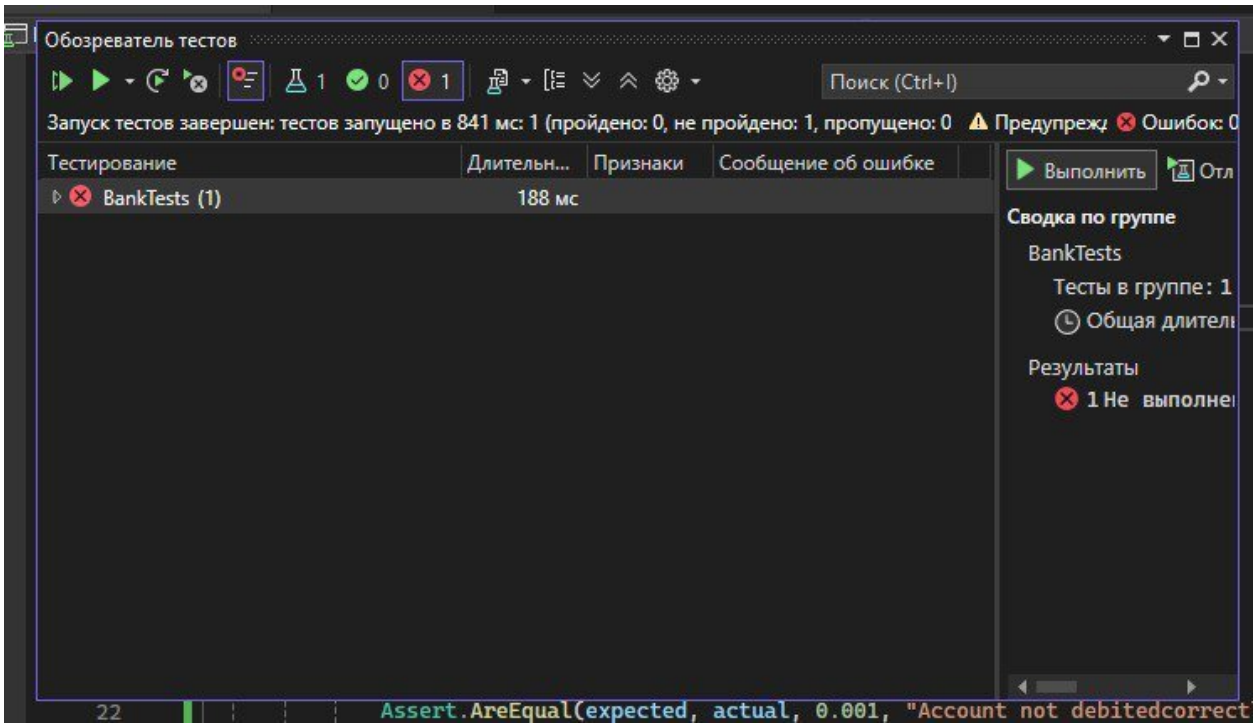

Рис.5 – Результат первого запуска

Шаг 4: Исправление и улучшение кода, повторный запуск тестов.

Код для BankAccount.cs:

```
using System;
namespace BankAccountNS
{
   /// <summary>
   /// Bank account demo class.
   /// </summary>
   public class BankAccount
   {
      private readonly string m_customerName;
      private double m_balance;
      public const string DebitAmountExceedsBalanceMessage = "Debit amount exceeds balance";
      public const string DebitAmountLessThanZeroMessage = "Debit amount is less than zero";
      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", amount,
DebitAmountExceedsBalanceMessage);
         }
         if (amount < 0)
         {
            throw new ArgumentOutOfRangeException("amount", amount,
DebitAmountLessThanZeroMessage);
         }
         m_balance -= amount;
      }
      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);
      }
   }
}
```

Код для BankAccountsTests.cs:

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
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 debitedcorrectly");
      }

      [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));
      }
      [TestMethod]
      public void Debit_WhenAmountIsMoreThanBalance_ShouldThrowArgumentOutOfRange()
      {
         // Arrange
         double beginningBalance = 11.99;
         double debitAmount = 20.00;
         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.");
      }
   }
}
```

После исправления повторно запускаем тесты и видим, что все тесты пройдены успешно. (Смотреть Рис.6)

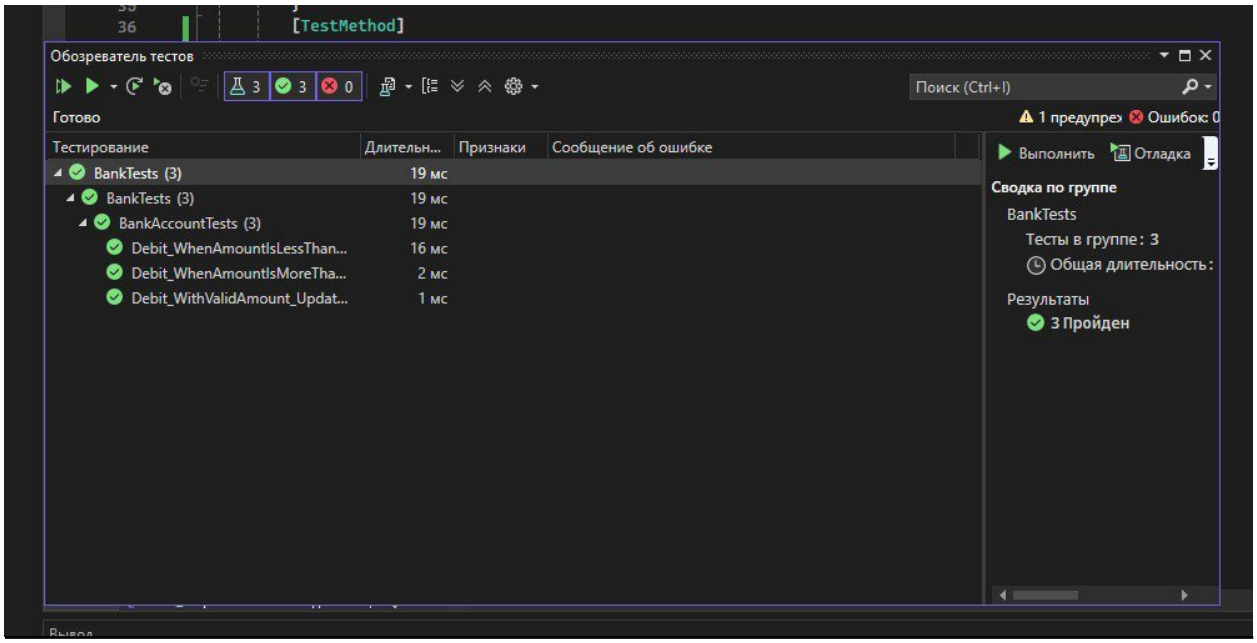

Рис.6 – Повторный запуск тестов

Вывод

В ходе работы были созданы модульные тесты для проверки метода Debit класса BankAccount. При первом запуске один тест не прошёл - это позволило обнаружить ошибку в коде. После исправления ошибки все три теста успешно выполнились.

Таким образом, модульное тестирование помогло найти и исправить ошибку, а также убедиться, что метод работает правильно. Цель практической работы достигнута.