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

Кафедра информационных технологий и программирования

Практическая работа #1

Тема работы: Создание и подключение базы данных

Работу выполнил:
Тырлов Ростислав Викторович
Группа: ИПс-11
Преподаватель: Берестов Дмитрий Борисович

Пермь 2025

Оглавление

Введение

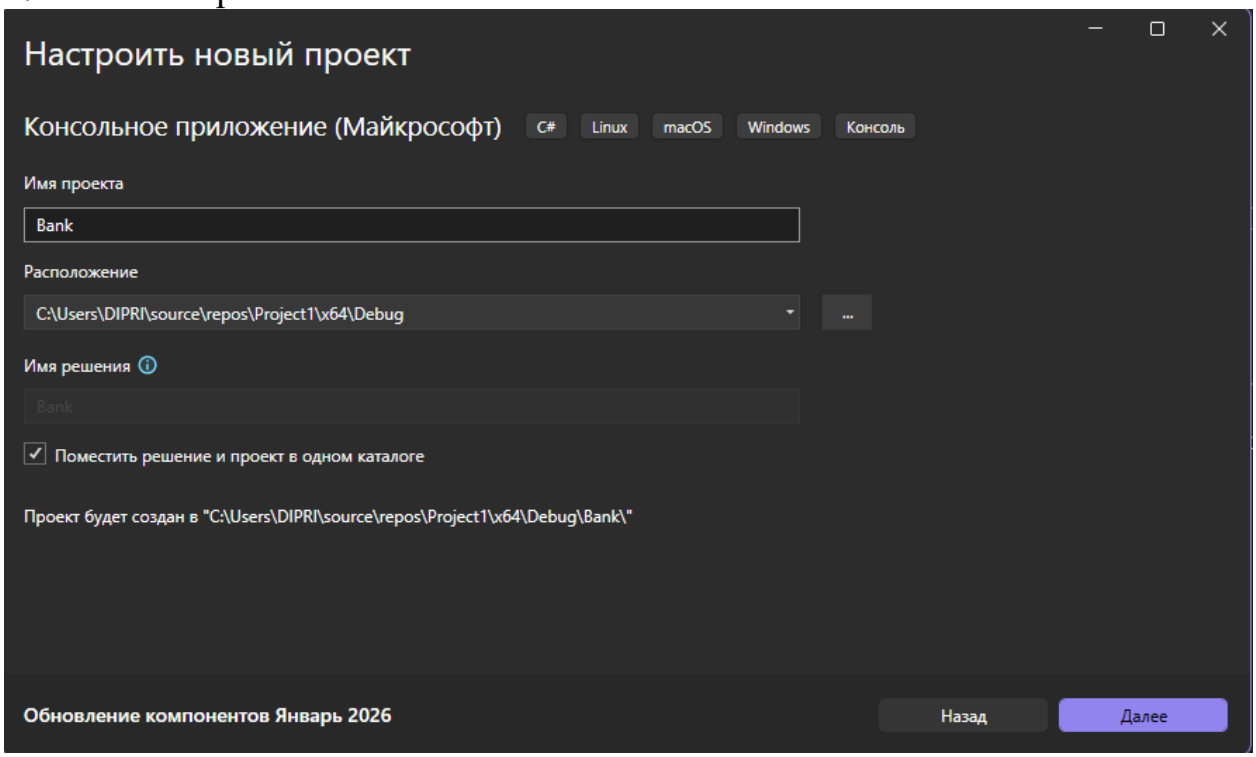
При выполнении практической №1 по теме “Средства тестирования Visual Studio-2022” я получил новый опыт в плане тестирования кода и др. информацию.

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

Microsoft Visual Studio — интегрированная среда разработки (IDE), созданная корпорацией Microsoft для профессионального программирования. Это комплексный набор инструментов, объединяющий редактор кода, компилятор, отладчик, инструменты анализа и многое другое в единой среде. Что было использовано в VS:

1. Консольное приложение (.NET Framework)
2. Проект модульного тестирования (.NET Framework)

Создание проекта:
1. Заходим в VS
2. Создаем проект
3. Выбираем из шаблона “Консольное приложение (.NET Framework)”
4. Называю проект “Bank”

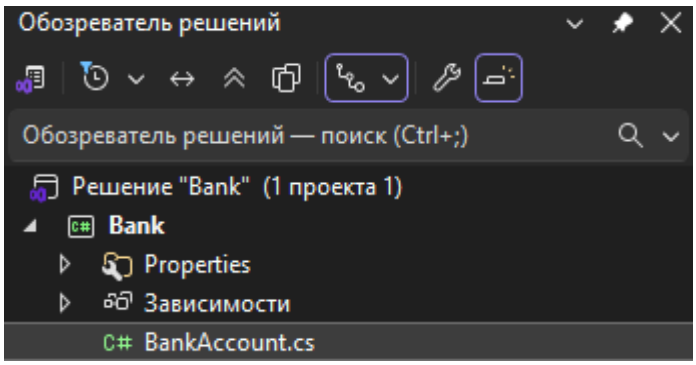


5. В “Program.cs” удаляем все и вставляем код который приложен в практической.

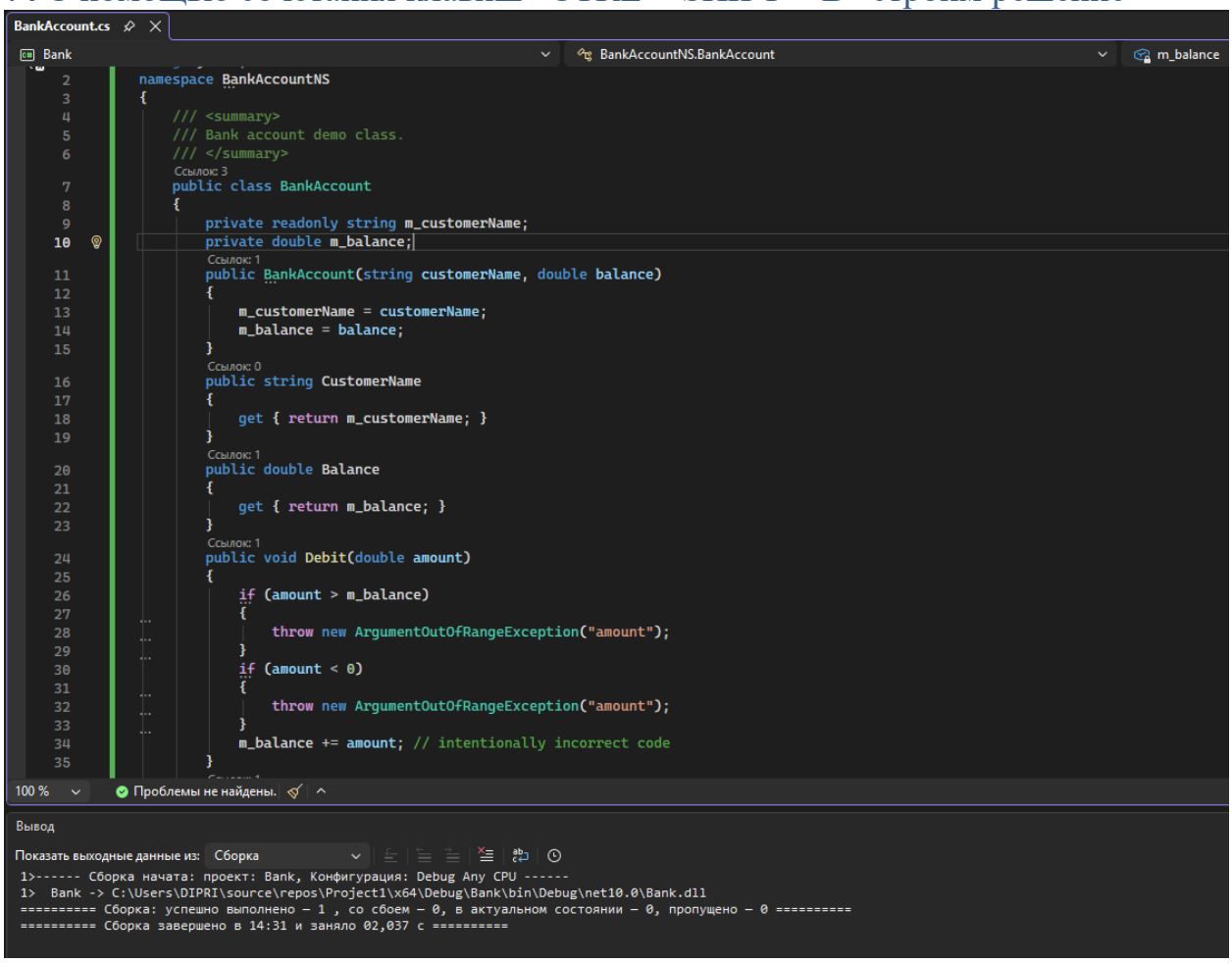
```
using System;
namespace BankAccountNS
{
   /// <summary>
   /// Bank account demo class.
   /// </summary>
   public class BankAccount
   {
      private readonly string m_customerName;
      private double m_balance;

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

            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);
        }
    }
}
```

6. Переименовываем “Program.cs” на “BankAccount.cs”

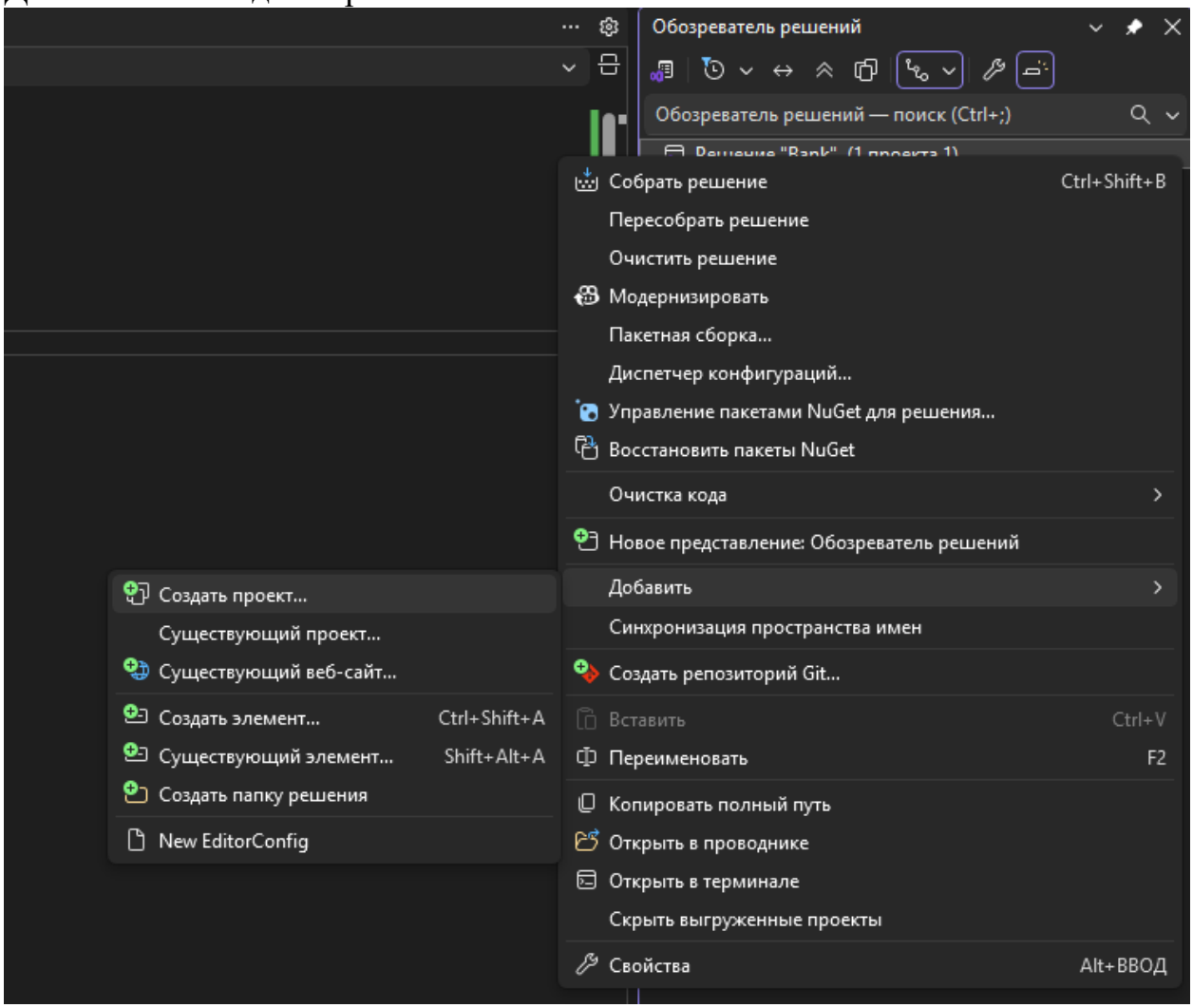


7. С помощью сочетания клавиш “CTRL + SHIFT + B” строим решение

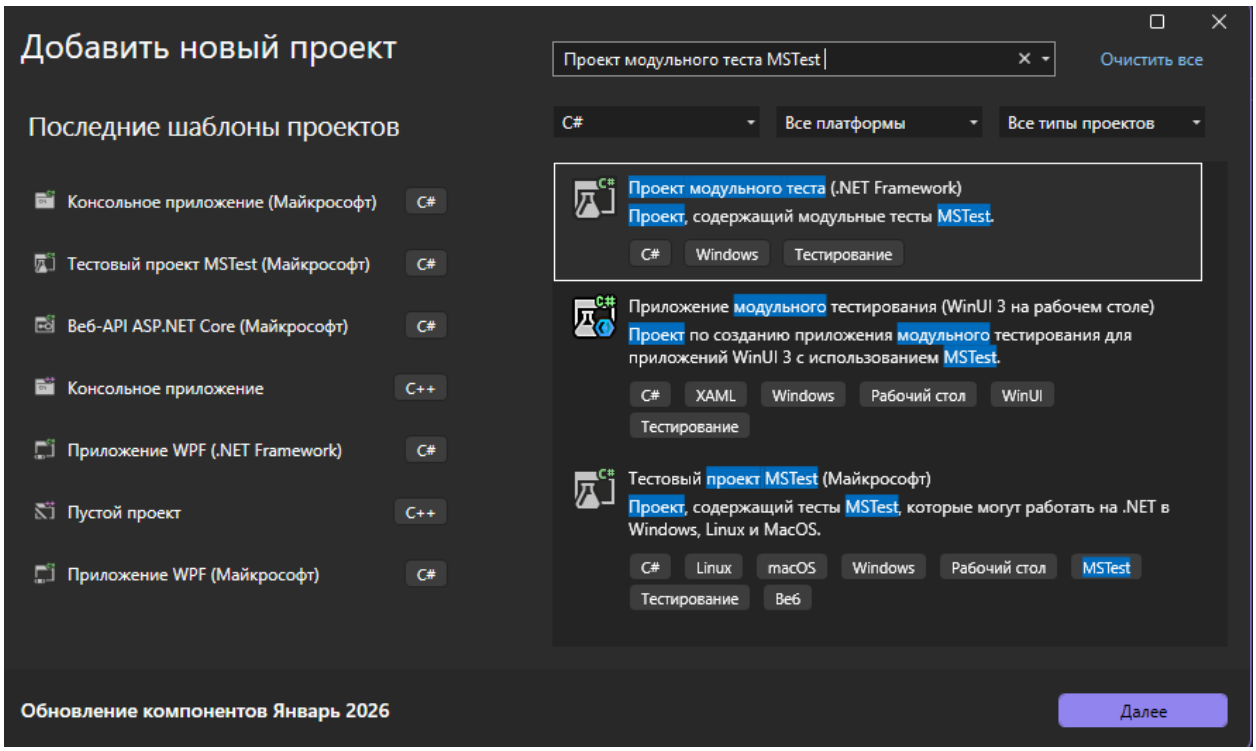


Создаем проект модульного теста

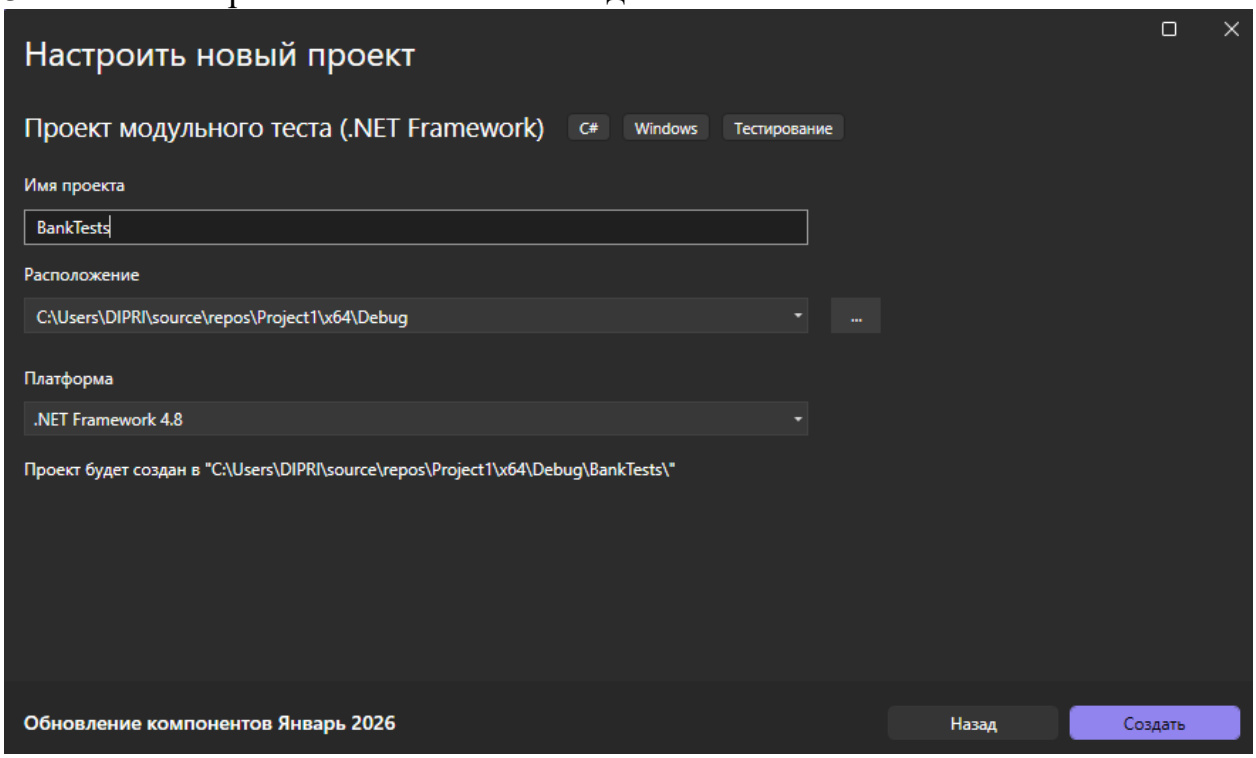
1. В обозревателе решений нажимаем “ПКМ” по “Решение “Bank”” -> Добавить -> Создать проект.



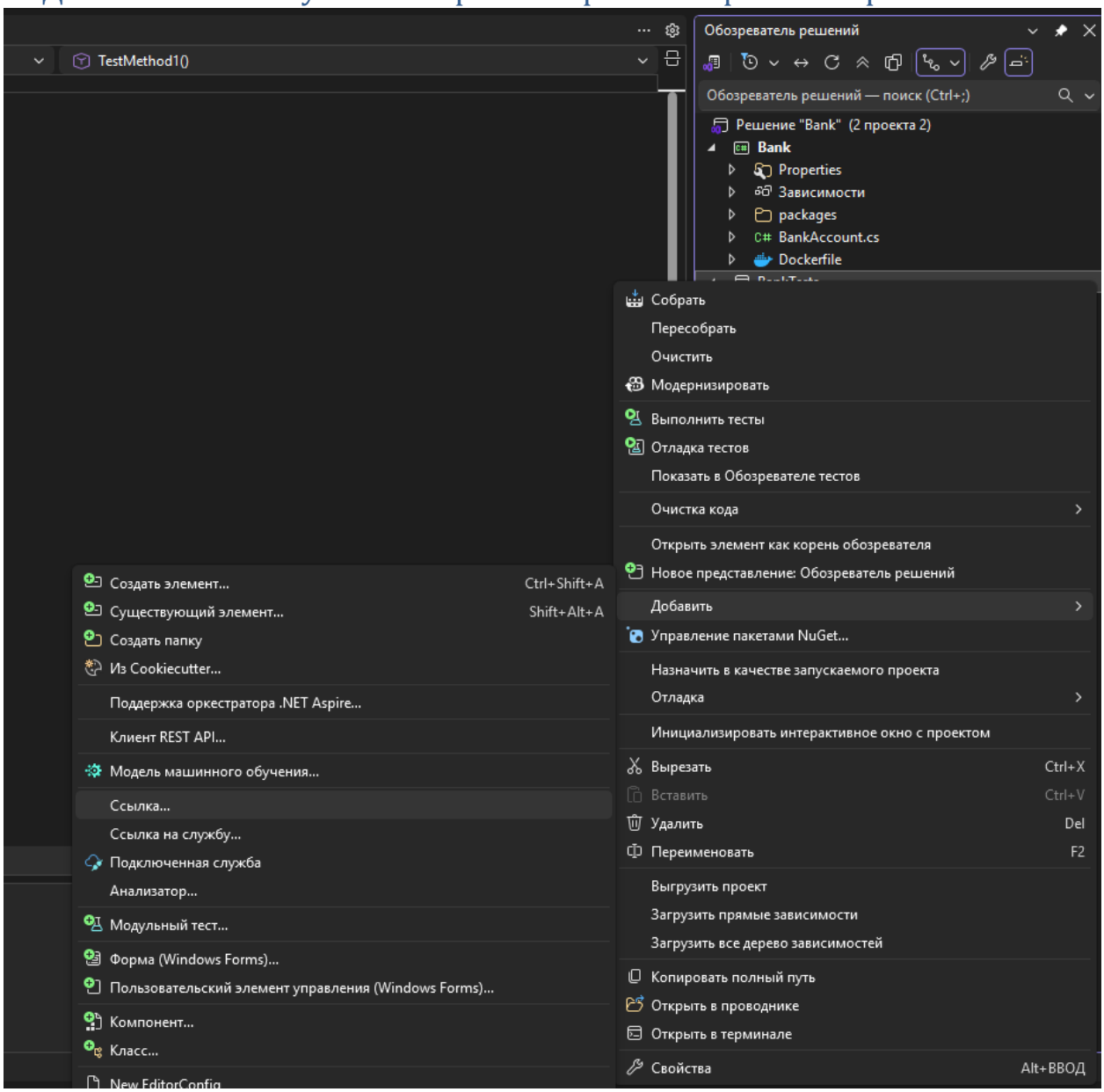
2. В поиске шаблона вписываем “Проект модульного теста MSTest” и выбираем “Проект модульного теста (.NET Framework)”



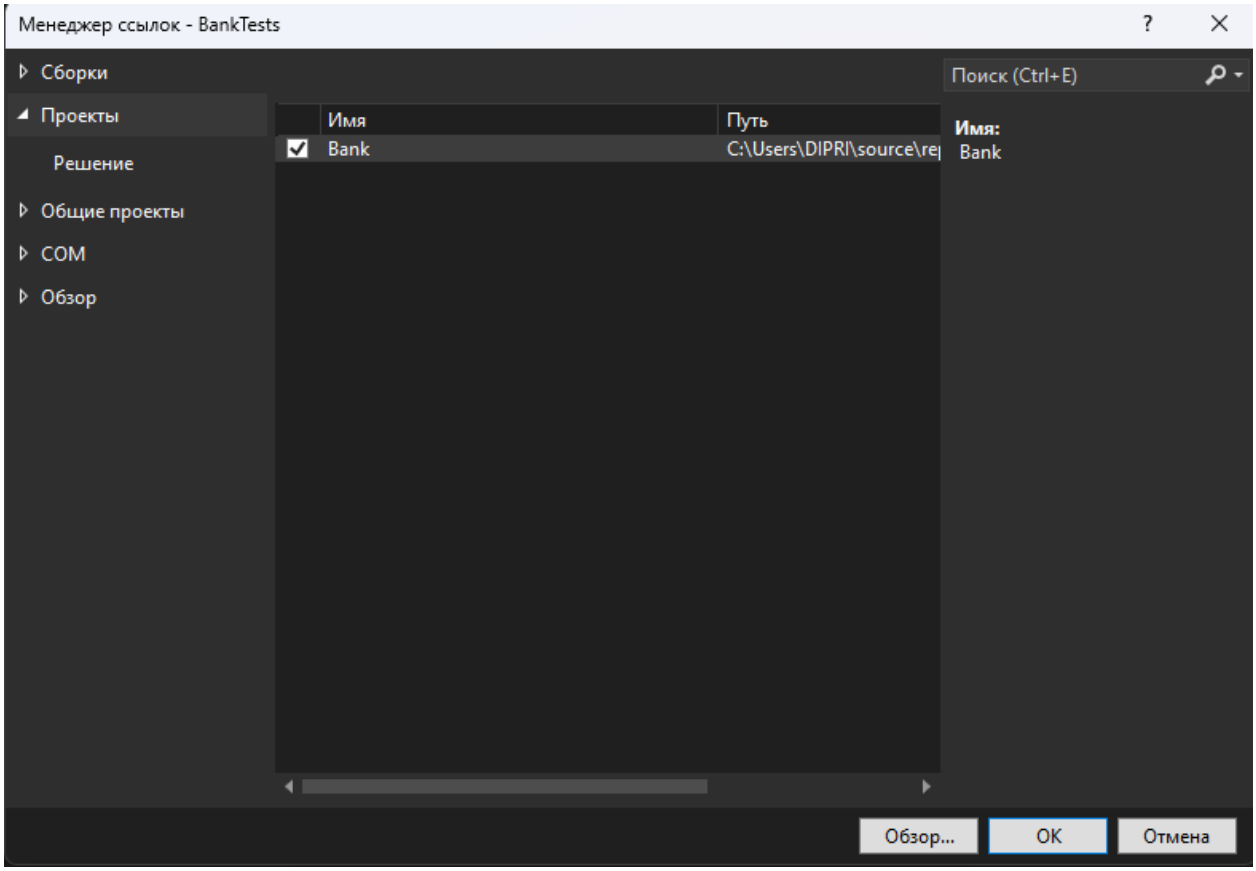
3. Называем проект “BankTests” и создаем



4. Добавляем ссылку на наш проект через “Обозреватель речений”

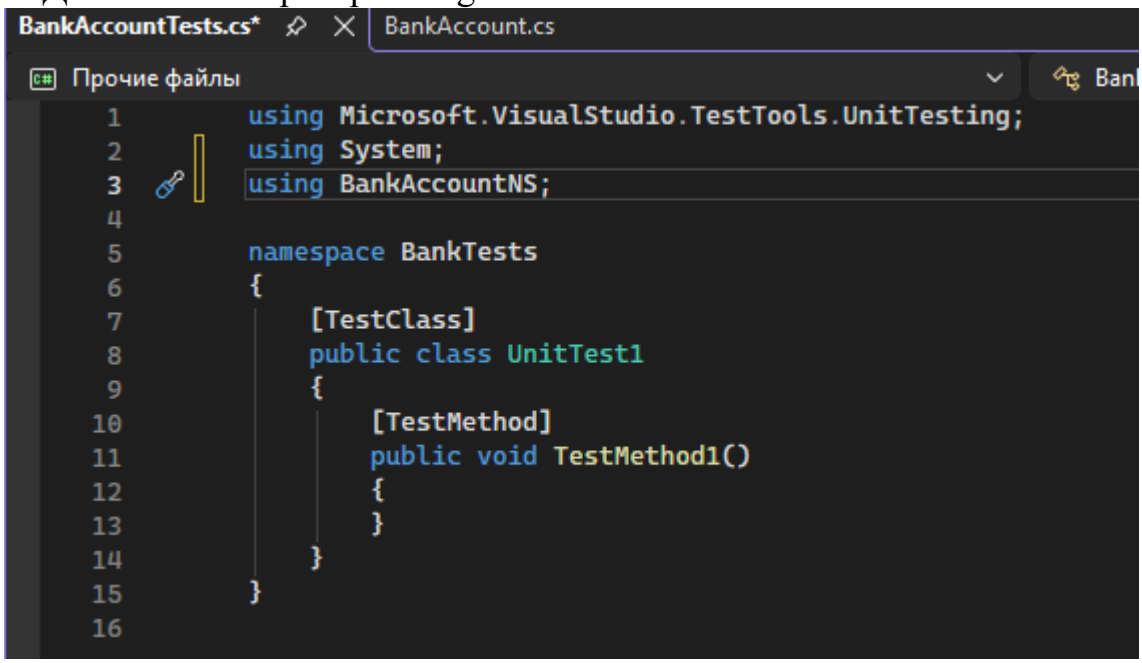


Ставим галочку на нашем проекте и нажимаем “ОК”

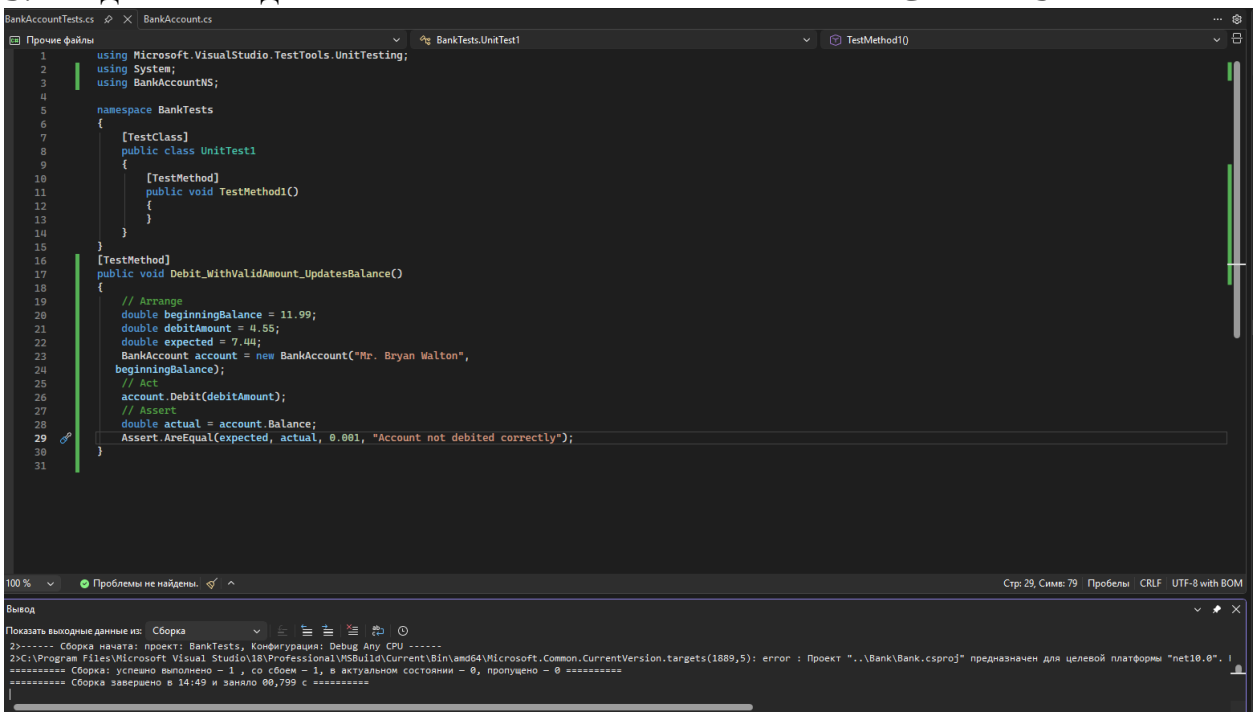

Менеджер ссылок - BankTests
Сборки
Проекты
Решение
Общие проекты
COM
Обзор
Поиск (Ctrl+E)
Имя
Путь
Bank
C:\Users\DIPRI\source\re
Имя:
Bank
Обзор...
OK
Отмена

Создание тестового класса

1. В “BankTests” переименовываем “UnitTest1.cs” на “BankAccountTests.cs”
2. Добавляем оператор “using”



3. Создаем метод теста и нажимаем сочетание клавиш “CTRL + SHIFT + B”



4. Запускаем тест

После запуска мы видим что вышли ошибки, дальше исправляем их
- m_balance += amount; (Меняем “+” на “-“)

5. Переписываем часть кода

```
using System;

namespace BankAccountNS
{
  public class BankAccount
  {
    public const string DebitAmountExceedsBalanceMessage = "Debit amount exceeds balance";
```

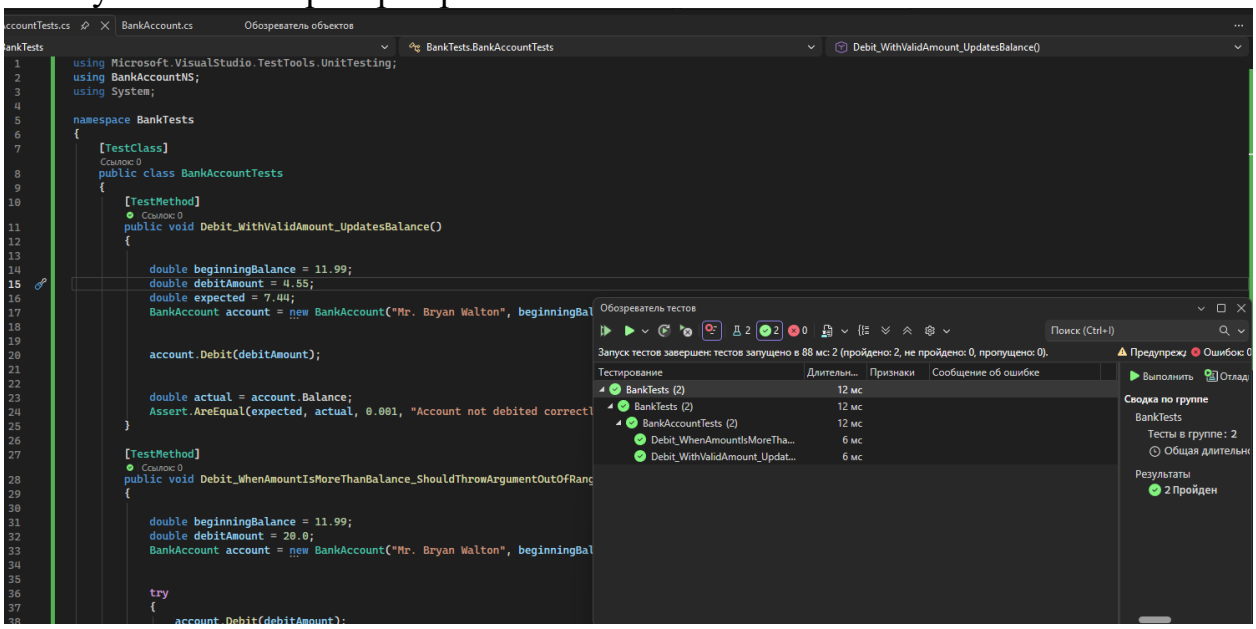
```
        public const string DebitAmountLessThanZeroMessage = "Debit amount is less than zero";

        private readonly string m_customerName;
        private double m_balance;

        public BankAccount(string customerName, double balance)
        {
            m_customerName = customerName;
            m_balance = balance;
        }

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

6. Создаем и запускаем новый метод теста

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

namespace BankTests
{
    [TestClass]
    public class BankAccountTests
    {
        [TestMethod]
        public void Debit_WithValidAmount_UpdatesBalance()
        {

            double beginningBalance = 11.99;
            double debitAmount = 4.55;
            double expected = 7.44;
            BankAccount account = new BankAccount("Mr. Bryan Walton", beginningBalance);


            account.Debit(debitAmount);


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

```
        [TestMethod]
        public void Debit_WhenAmountIsMoreThanBalance_ShouldThrowArgumentOutOfRange()
        {

            double beginningBalance = 11.99;
            double debitAmount = 20.0;
            BankAccount account = new BankAccount("Mr. Bryan Walton", beginningBalance);


            try
            {
                account.Debit(debitAmount);
            }
            catch (ArgumentOutOfRangeException e)
            {
                StringAssert.Contains(e.Message, BankAccount.DebitAmountExceedsBalanceMessage);
                return;
            }

            Assert.Fail("The expected exception was not thrown.");
        }
    }
}
```

И запускаем тест проверяя работает ли все или нет.



Все работает.

Вывод:

Это было сложно, потому что не было подробно расписано что и где открывать. К примеру возьмем создание проекта модульного теста (.CORE), его попросту нету в списке шаблонов и пришлось искать в ручную без каких либо знаний. Но я все равно благодарен за этот опыт, думаю со временем привыкну и дальше будет проще.