

Министерство науки и высшего образования Российской Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ  
В.Ф.УТКИНА»  
Рязанский станкостроительный колледж РГРТУ

Лабораторная работа 36

**Дополнительная профессиональная программа  
повышения квалификации  
«Тестирование программного обеспечения (с учетом стандарта Ворлдскиллс по  
компетенции «Программные решения для бизнеса»)»**

Мазуренко Валерия Витальевна

Рязань 2022

## Практическая работа №37

### Модульное тестирование.

#### Тестирование классов

Цель работы: Изучение основ модульного тестирования, тестирование классов

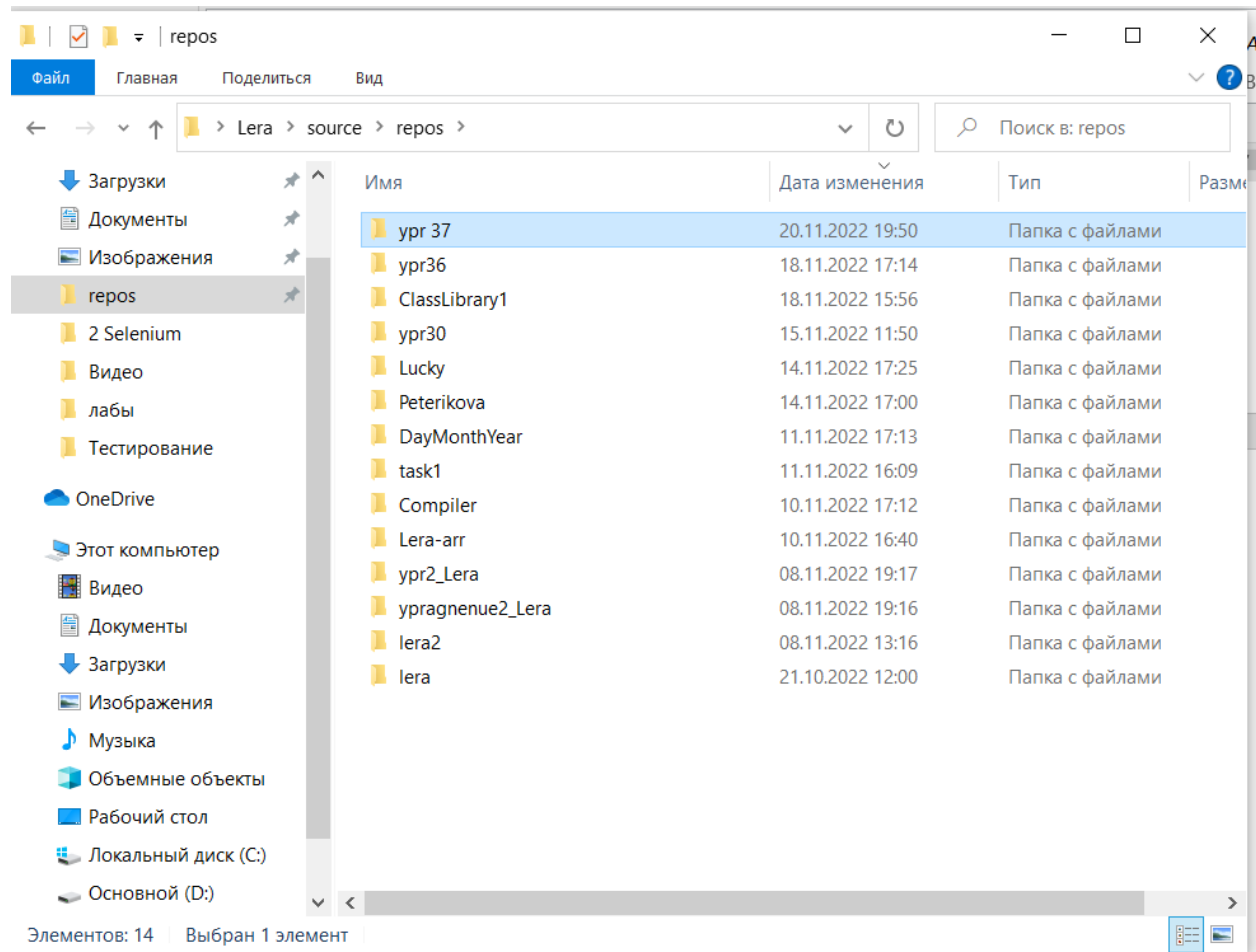
Задание:

1. Из папки Копия скопировать проект Bank себе в рабочую область.
2. Открыть проект Bank и запустить файл BankAccount.cs (в папке Bank).
3. Следуя инструкции, создание проекта модульного теста.
4. Следуя инструкции, создание тестового класса. 5. Следуя инструкции, создание метода теста.
6. Произвести сборку и запуск теста.
7. Исправление кода и повторный запуск тестов.
8. В отчет включить архив проекта, код (листинг) программы и скриншоты выполненных тестов.

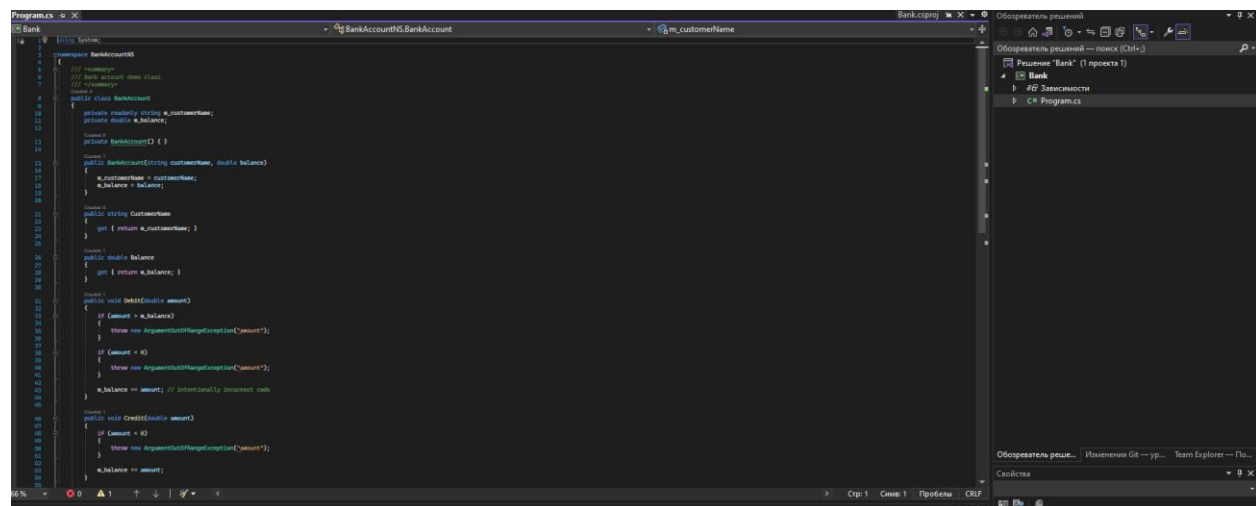
## Лабораторная работа №37

### Модульное тестирование.

Из папки Копия скопировать проект Bank себе в рабочую область.

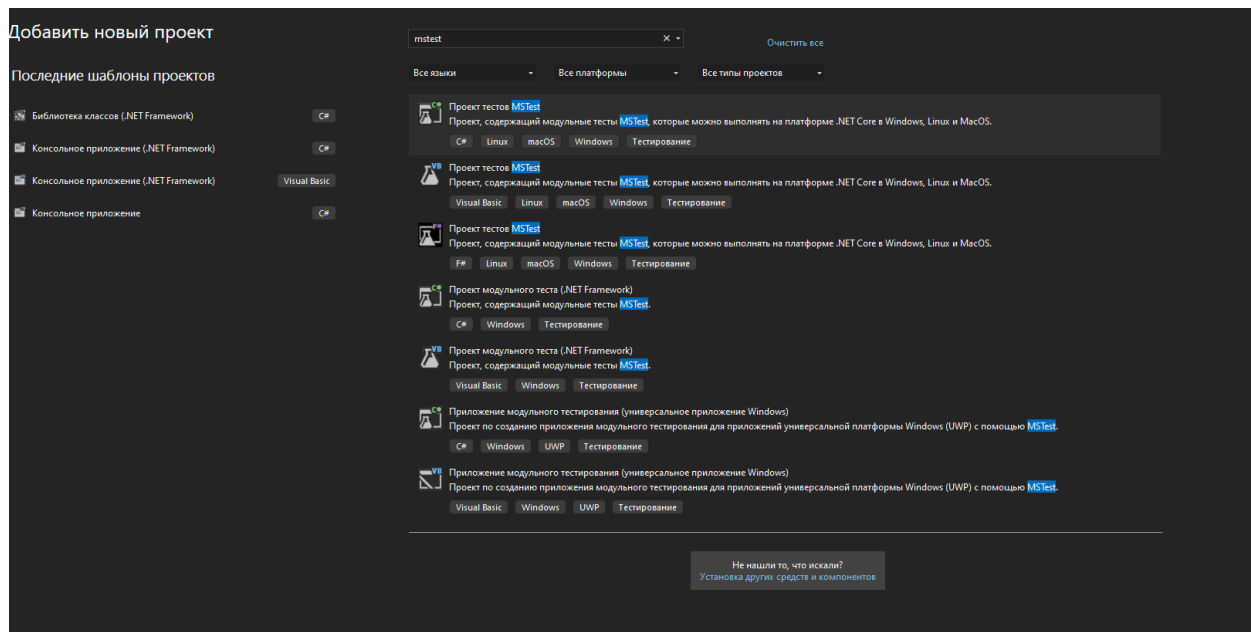


Открыть проект Bank и запустить файл BankAccount.cs (в папке Bank).



### Создание проекта модульного теста.

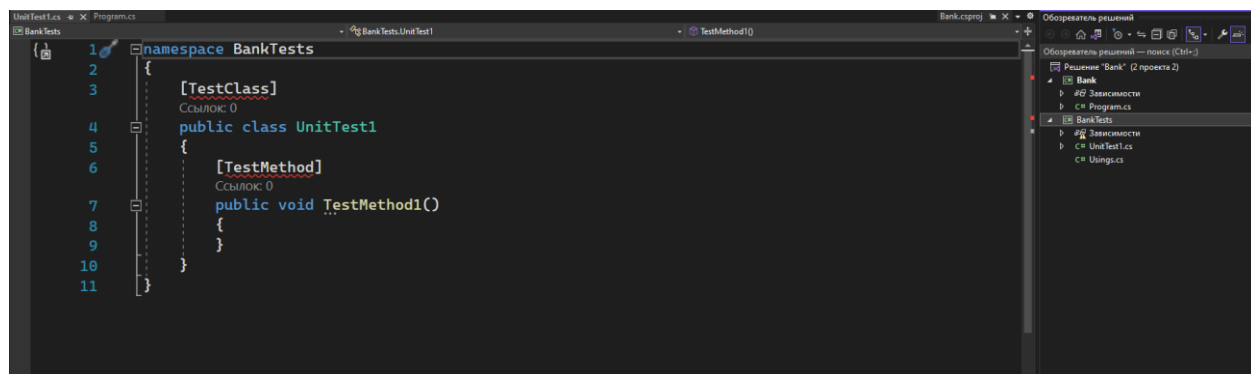
В обозревателе решений щелкните решение правой кнопкой мыши и выберите пункты Добавить>Создать проект. Введите test в поле поиска, выберите C# в качестве языка, затем выберите Проект модульного теста MSTest (.NET Core) для C# в качестве шаблона .NET Core и щелкните Далее.



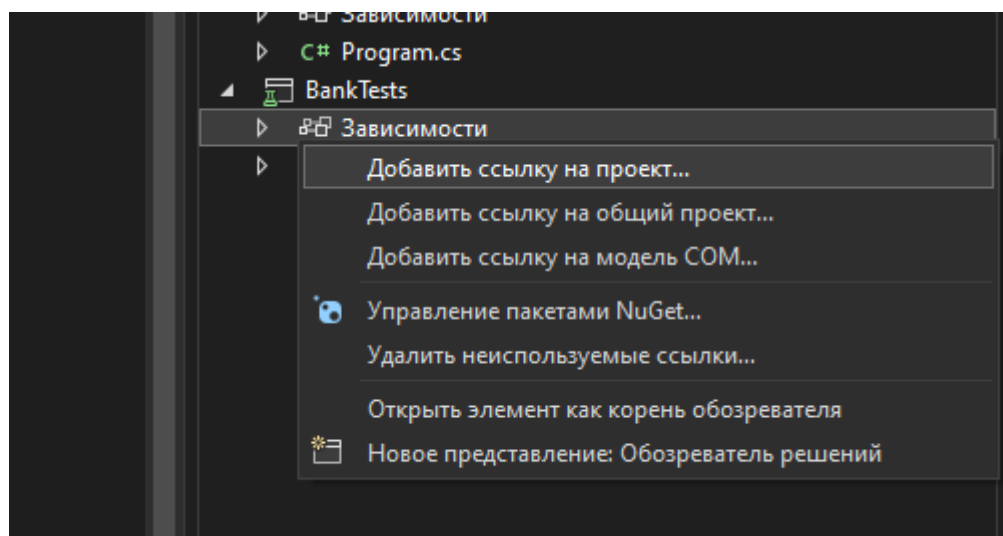
Назовите проект BankTests и щелкните Далее.

Выберите рекомендуемую версию целевой платформы или .NET 6 и щелкните Создать.

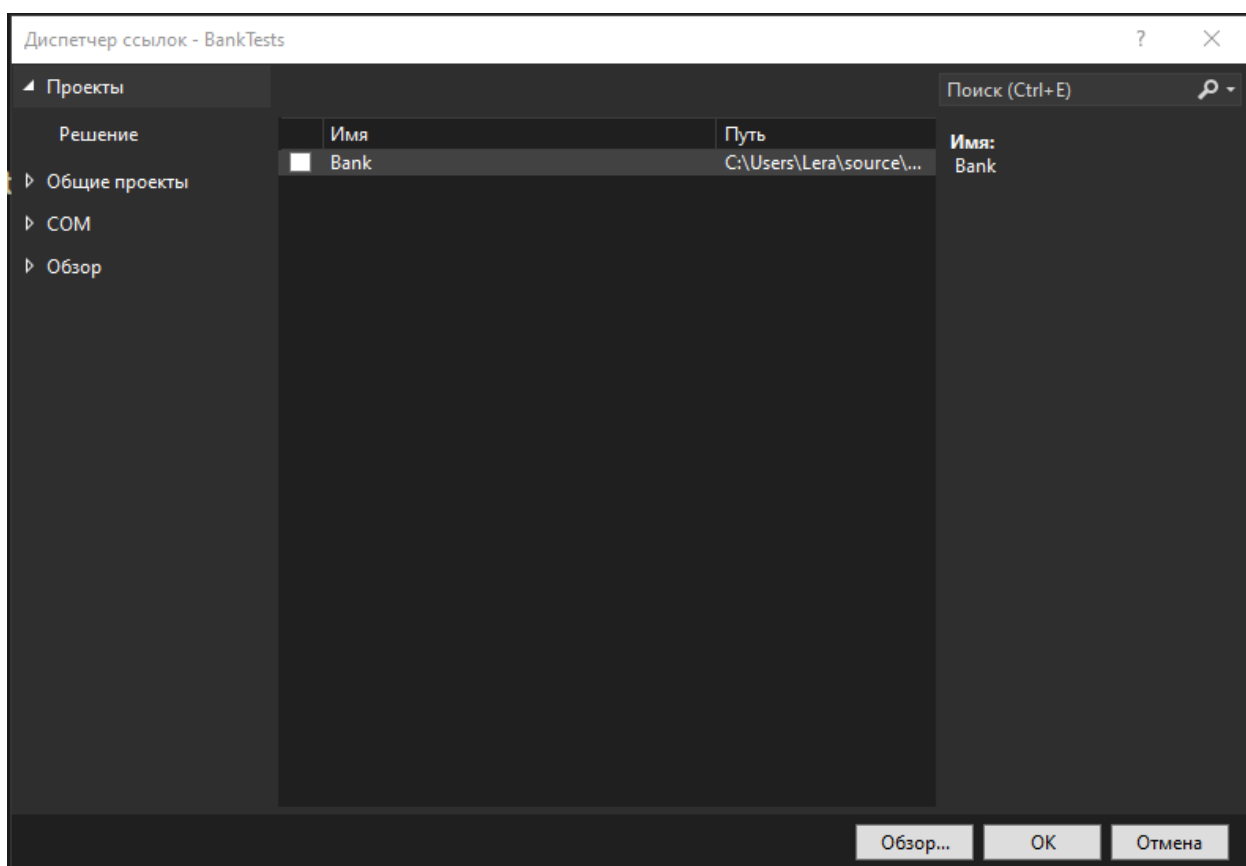
Проект BankTests добавляется в решение Банк



В проекте BankTests добавьте ссылку на проект Банк. В обозревателе решений щелкните Зависимости в проекте BankTests, а затем выберите в контекстном меню элемент Добавить ссылку или Добавить ссылку на проект.



В диалоговом окне Диспетчер ссылок разверните Проекты, выберите Решение и выберите элемент Банк.



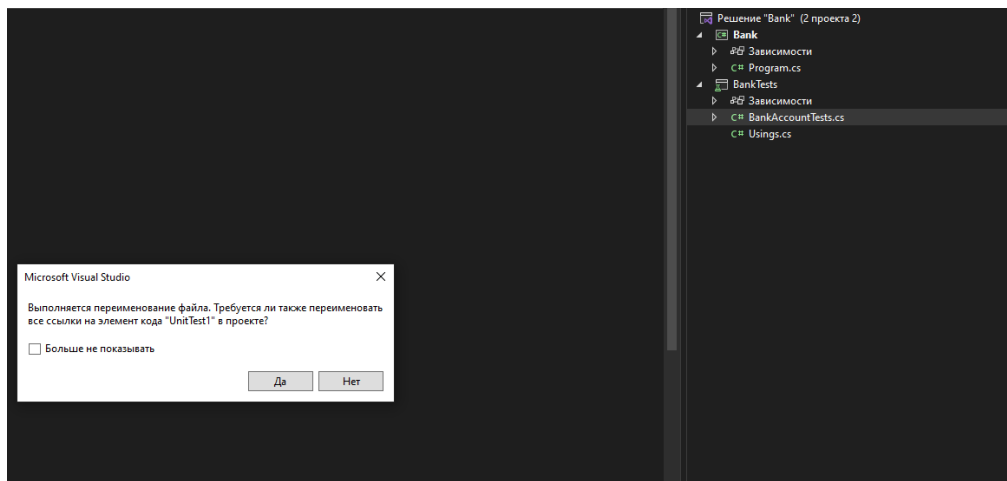
### Создание тестового класса

Создание тестового класса, чтобы проверить класс BankAccount. Можно использовать UnitTest1.cs, созданный в шаблоне проекта, но лучше дать файлу и классу более описательные имена.

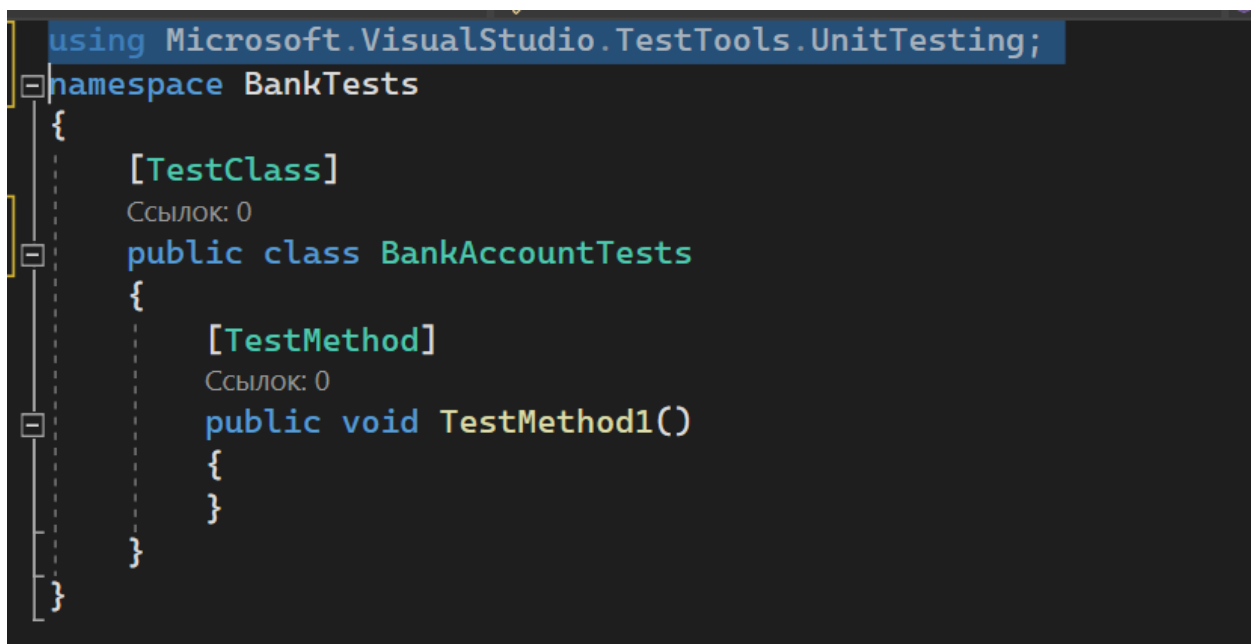
Переименуйте файл и класс

1. Чтобы переименовать файл, в обозревателе решений выберите файл UnitTest1.cs в проекте BankTests. В контекстном меню выберите команду Переименовать (или нажмите клавишу F2), а затем переименуйте файл в BankAccountTests.cs.

2. Чтобы переименовать класс, поместите курсор в UnitTest1 в редакторе кода, щелкните правой кнопкой мыши и выберите команду Переименовать (или нажмите клавиши F2). Введите название BankAccountTests и нажмите клавишу ВВОД.

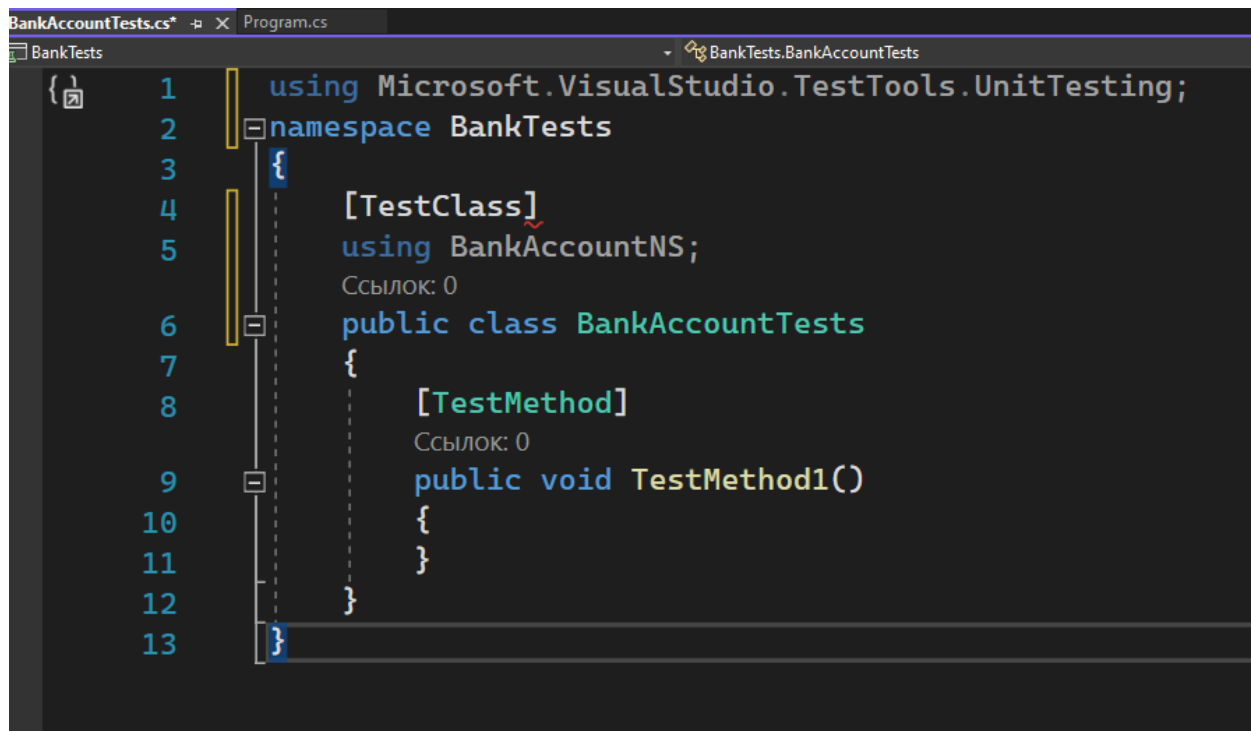


Файл BankAccountTests.cs теперь содержит следующий код: using Microsoft.VisualStudio.TestTools.UnitTesting



Добавьте оператор using Можно также добавить оператор using в класс, чтобы тестируемый проект можно было вызывать без использования полных имен. Вверху файла класса добавьте: using Microsoft.VisualStudio.TestTools.UnitTesting;

```
namespace BankTests {  
    [TestClass]  
    using BankAccountNS;  
    public class UnitTest1
```



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

### Требования к тестовому классу

Минимальные требования к тестовому классу следующие:

- Атрибут [TestClass] является обязательным в любом классе, содержащем методы модульных тестов, которые необходимо выполнить в обозревателе тестов.
- Каждый метод теста, предназначенный для запуска в обозревателе тестов, должен иметь атрибут [TestMethod].

Можно иметь другие классы в проекте модульного теста, которые не содержат атрибута [TestClass], а также иметь другие методы в тестовых классах, у которых атрибут — [TestMethod] .

Можно вызывать эти другие классы и методы в методах теста.

### Создание первого тестового метода

В этой процедуре мы напомним методы модульного теста для проверки поведения метода Debit класса BankAccount.

Существует по крайней мере три поведения, которые требуется проверить:

- Метод создает исключение ArgumentOutOfRangeException , если сумма по дебету превышает баланс.
- Метод создает исключение ArgumentOutOfRangeException, если сумма по дебету меньше нуля.
- Если значение дебета допустимо, то метод вычитает сумму дебета из баланса счета

Создание метода теста Первый тест проверяет, снимается ли со счета нужная сумма при допустимом размере кредита (со значением меньшим, чем баланс счета, и большим, чем ноль).

Добавьте следующий метод в этот класс BankAccountTests :

### Создание метода теста

Первый тест проверяет, снимается ли со счета нужная сумма при допустимом размере кредита (со значением меньшим, чем баланс счета, и большим, чем ноль).

Добавьте следующий метод в этот класс

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

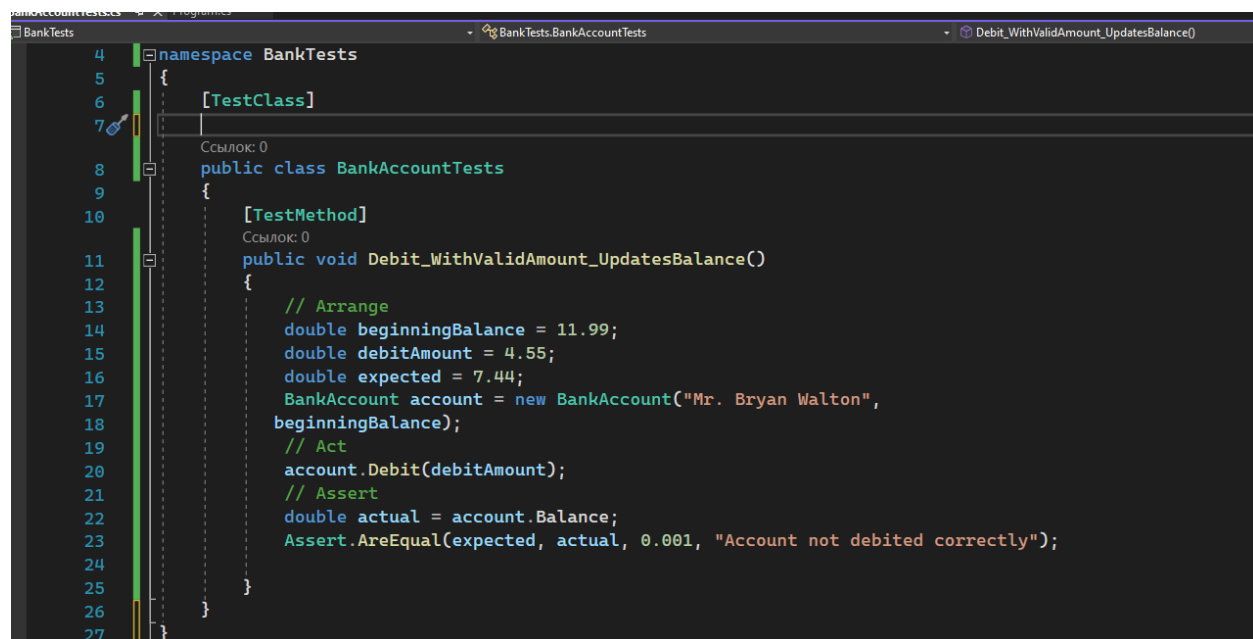
account.Debit(debitAmount);

// Assert

double actual = account.Balance;

Assert.AreEqual(expected, actual, 0.001, "Account not debited correctly");

}



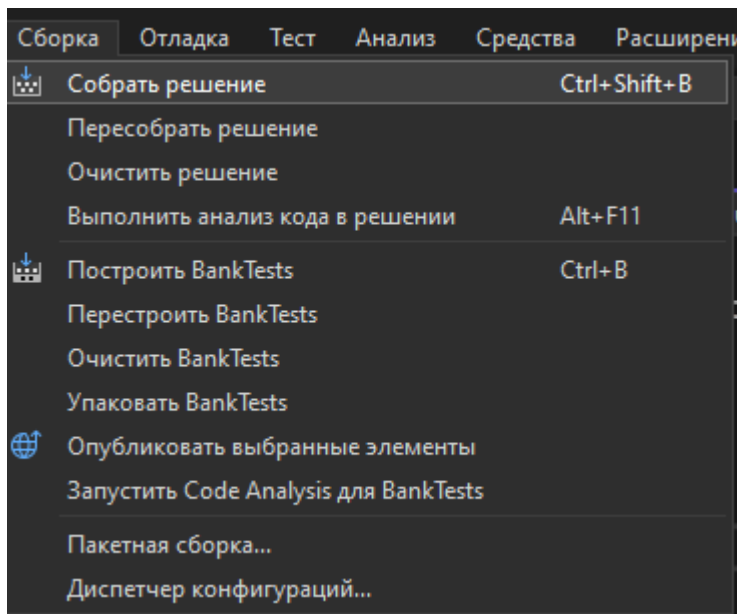
Метод очень прост: он создает новый объект BankAccount с начальным балансом, а затем снимает допустимое значение. Он использует метод Assert.AreEqual, чтобы проверить, что конечный баланс соответствует ожидаемому. Такие методы, как Assert.AreEqual, Assert.IsTrue и другие, зачастую используются в модульном тестировании.

Дополнительную концептуальную информацию о написании модульного теста см. в разделе Написание тестов.

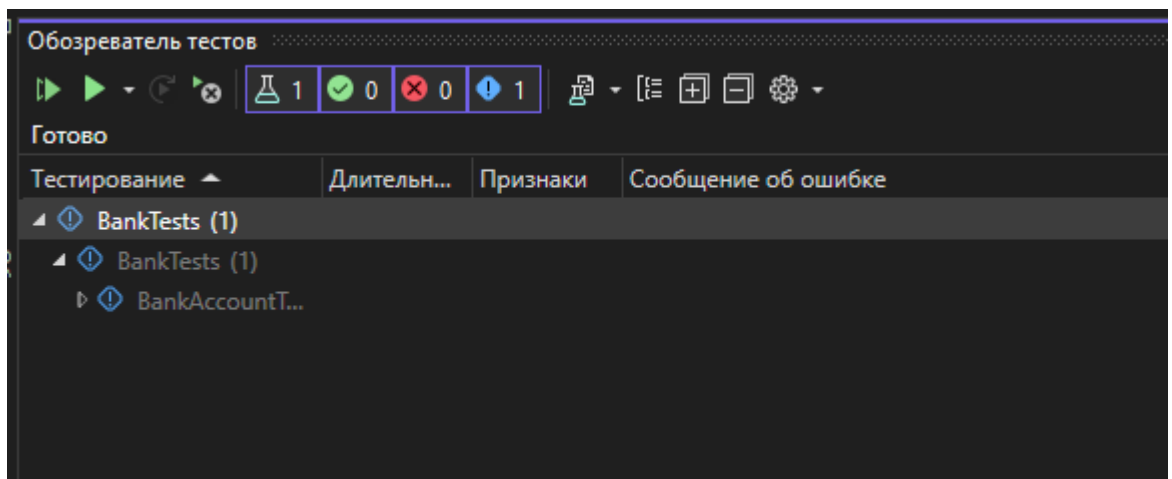
## Сборка и запуск теста



1. В меню Сборка нажмите Построить решение (или нажмите клавиши CTRL + SHIFT + B).



2. Откройте Обзоратель тестов, выбрав Тест>Windows>Обзоратель тестов в верхней строке меню (или нажмите клавиши CTRL + E, T).



3. Выберите Запустить все, чтобы выполнить тест (или нажмите клавиши CTRL + R, V).

Во время выполнения теста в верхней части окна Обзоратель тестов отображается анимированная строка состояния. По завершении тестового запуска строка состояния становится зеленой, если все методы теста успешно пройдены, или красной, если какие-либо из тестов не пройдены. В данном случае тест пройден не будет.

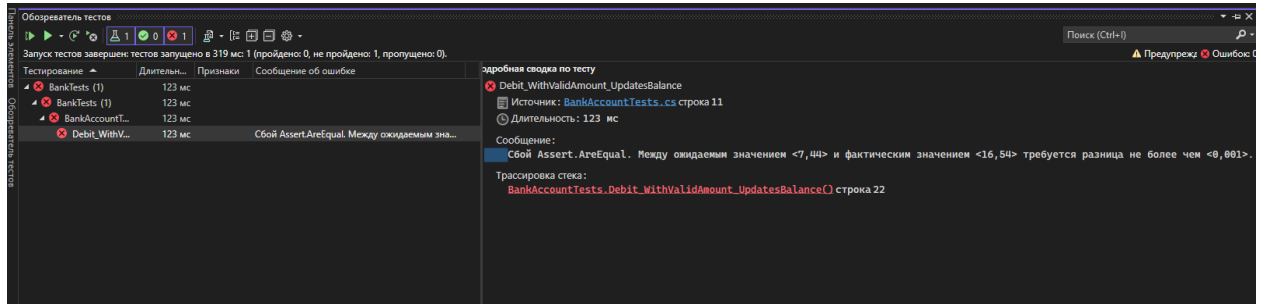
4. Выберите этот метод в обзорателе тестов для просмотра сведений в нижней части окна.  
Debit\_WithValidAmount\_UpdatesBalance  
Источник: BankAccountTests.cs строка 11  
Длительность: 123 мс

Сообщение:

Сбой Assert.AreEqual. Между ожидаемым значением <7,44> и фактическим значением <16,54> требуется разница не более чем <0,001>. Account not debited correctly

Трассировка стека:

BankAccountTests.Debit\_WithValidAmount\_UpdatesBalance() строка 22



В данном случае тест пройден не будет.

## Исправление кода и повторный запуск тестов

Результат теста содержит сообщение, описывающее возникшую ошибку. Чтобы увидеть это сообщение, может потребоваться выполнить детализацию. Для метода `AreEqual` выводится сообщение о том, что ожидалось и что было фактически получено. Ожидалось, что баланс уменьшится, а вместо этого он увеличился на сумму списания.

Модульный тест обнаружил ошибку: сумма списания добавляется на баланс счета, вместо того чтобы вычитаться.

Исправление ошибки

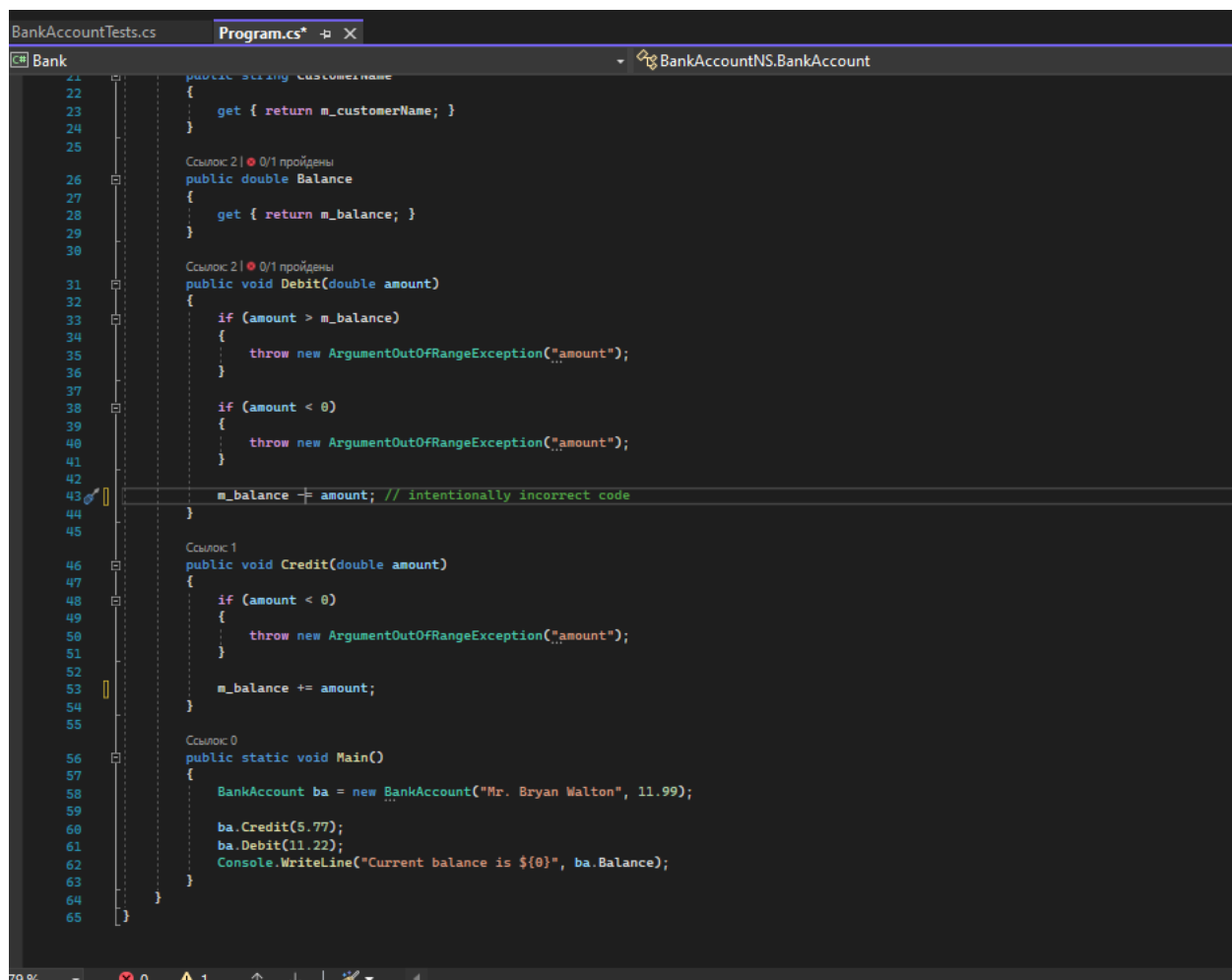
Чтобы исправить эту ошибку, в файле `BankAccount.cs` замените строку:

```
m_balance += amount;
```

на:

C#Копировать

```
m_balance -= amount;
```



```
BankAccountTests.cs Program.cs* X
Bank
21 public string CustomerName
22 {
23     get { return m_customerName; }
24 }
25
26 Ссылка 2 | 0/1 пройдены
27 public double Balance
28 {
29     get { return m_balance; }
30 }
31
32 Ссылка 2 | 0/1 пройдены
33 public void Debit(double amount)
34 {
35     if (amount > m_balance)
36     {
37         throw new ArgumentOutOfRangeException("amount");
38     }
39
40     if (amount < 0)
41     {
42         throw new ArgumentOutOfRangeException("amount");
43     }
44
45     m_balance += amount; // intentionally incorrect code
46 }
47
48 Ссылка 1
49 public void Credit(double amount)
50 {
51     if (amount < 0)
52     {
53         throw new ArgumentOutOfRangeException("amount");
54     }
55
56     m_balance += amount;
57 }
58
59 Ссылка 0
60 public static void Main()
61 {
62     BankAccount ba = new BankAccount("Mr. Bryan Walton", 11.99);
63     ba.Credit(5.77);
64     ba.Debit(11.22);
65     Console.WriteLine("Current balance is ${0}", ba.Balance);
66 }
```

Повторный запуск теста

В обозревателе тестов выберите Запустить все, чтобы запустить тест повторно (или нажмите клавиши `CTRL + R, V`). Красно-зеленая строка становится зеленой, чтобы указать, что тест был пройден

Обозреватель тестов

▶

⏮

⏪

⏩

⏭

⏴

⏵

⏶

⏷

⏸

⏹

⏺

⏻

⏼

⏽

⏾

⏿

⏺

⏻

⏼

⏽

⏾

⏿

110

10

Поиск (Ctrl+I)

Запуск тестов завершен: тестов запущено в 208 мс: 1 (пройдено: 1, не пройдено: 0, пропущено: 0).

Тестирование

Длительность

Признаки

Сообщение об ошибке

BankTests (1)

36 мс

BankTests (1)

36 мс

BankAccountT...

36 мс

Debit\_WithV...

36 мс

Подробная сводка по тесту

Debit\_WithValidAmount\_UpdatesBalance

Источник: BankAccountTests.cs строка 11

Длительность: 36 мс