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

Отделение информационных технологий и программирования

Практическая работа #1

Тема работы: “Модульное тестирование в Visual Studio”

Работу выполнил: Лазуков Всеволод Николаевич
Группа: ИПо-21
Преподаватель: Берестов Дмитрий Борисович

Пермь 2026

Оглавление

1. Цель и задачи работы

В ходе выполнения данной практической работы были освоены принципы создания модульных тестов для управляемого кода в среде .NET. Ключевыми задачами являлись: получение навыков написания автотестов с применением фреймворка MSTest, выявление программных дефектов посредством тестирования, а также практическая отработка методологии TDD (разработка через тестирование).

2. Объект тестирования

Объектом исследования выступил класс BankAccount, имитирующий работу банковского счета. Класс включает свойства для хранения имени владельца и текущего баланса, а также методы Credit() (пополнение) и Debit() (списание). Основное внимание было уделено методу Debit(). В исходной реализации метода Debit() присутствовала намеренная логическая ошибка: вместо операции вычитания суммы выполнялось её прибавление к балансу. Функционал тестов был направлен на обнаружение данной ошибки и верификацию корректности работы после внесения исправлений.

3. Структура решения

Работа выполнялась в среде Visual Studio. Решение состояло из двух проектов, целевой платформой для обоих выбран .NET 6: Bank — библиотека классов, содержащая тестируемую бизнес-логику. BankTests — проект модульных тестов, созданный на базе MSTest.

В проекте тестов была подключена зависимость MSTest.TestFramework версии 4.0.1 и настроена ссылка на проект Bank, что обеспечило доступ тестов к классам основной библиотеки.

4. Ход выполнения тестирования

4.1. Проверка базовой функциональности

Сначала был разработан тест для проверки штатного сценария: списание корректной суммы должно приводить к уменьшению баланса. Ниже приведен код теста с использованием конкретных переменных:

```
[TestMethod]
public void Debit_WithValidAmount_UpdatesBalance()
{
        double beginningBalance = 11.99;
        double debitAmount = 4.55;
        double expected = 7.44;
        var account = new BankAccount("Mr. Bryan Walton", beginningBalance);
        account.Debit(debitAmount);
        Assert.AreEqual(expected, account.Balance, 0.001);
}
```

При первоначальном запуске тест завершился неудачей: фактический баланс составил 16.54 вместо ожидаемых 7.44.

4.2. Исправление дефекта

Анализ кода выявил ошибочную операцию присваивания. Было : *m_balance += amount*. Стало : *m_balance -= amount*; . После замены оператора сложения на вычитание тест успешно прошел.

4.3. Тестирование граничных условий

Для обеспечения надежности метода были добавлены проверки на некорректные входные данные. Реализованы тесты для ситуаций, когда сумма списания отрицательная или превышает доступный баланс. В обоих случаях метод должен выбрасывать исключение *ArgumentOutOfRangeException*. Для валидации использована конструкция *try/catch* и проверка сообщения исключения:

```
[TestMethod]
public void
Debit_WhenAmountIsLessThanZero_ShouldThrowArgumentOutOfRange()
{
    double beginningBalance = 11.99;
    double debitAmount = -100.00;
    BankAccount account = new BankAccount("Mr. Bryan Walton",
    beginningBalance);
    try
    {
        account.Debit(debitAmount);
        Assert.Fail("Исключение не было выброшено");
    }
    catch (ArgumentOutOfRangeException e)
    {
        StringAssert.Contains(e.Message,
BankAccount.DebitAmountLessThanZeroMessage);
    }
}
```

5. Результаты работы

По итогам выполнения работы класс *BankAccount* реализует корректную логику финансовых операций. Разработано три модульных теста, покрывающих следующие сценарии:

1. Успешное списание средств.
2. Обработка отрицательного значения аргумента.
3. Обработка попытки списания суммы, превышающей баланс.

Все тесты успешно проходят в обозревателе тестов Visual Studio.

6. Заключение

В ходе практики были работы с тестовыми проектами в Visual Studio и фреймворком MSTest. Освоено использование атрибутов *[TestClass]* и *[TestMethod]*, утверждений *Assert.AreEqual*, *Assert.ThrowsException*, *StringAssert.Contains*. Получен опыт отладки кода на основе результатов тестов и проведения рефакторинга без нарушения работоспособности системы. Внедрение модульного тестирования позволило автоматически обнаружить ошибку в методе Debit и создало защитный механизм от регрессий в будущем.