МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра МСС

Гилевич Павел Геннадьевич

«Unit-тестирование»

Отчет по лабораторной работе
студента 3 курса 12 группы
по дисциплине «Технологии программирования»

Преподаватель
Довнар С.Е.

Минск 2013

После написания приложения для банковской системы в среде Visual Studio были созданы тесты для некоторых методов классов проекта, в которых была проверена правильность работы данных методов. Приведем примеры трех тестов.

1. Метод *WithdrawMoney* класса *TimedMaturityAccount*

```
public void WithdrawMoneyTest()
        {
            int number = 2;
            string password = string.Empty;
            double balance = 100;
            double rate = 36;
            DateTime openDate = new DateTime(2013, 1, 1);
            double penalty_rate = 10;
            int period = 2;
            TimedMaturityAccount target = new TimedMaturityAccount(number, password, balance, rate,
openDate, penalty_rate, period);
            DateTime currentDate = new DateTime(2013, 2, 10);
            double expected = 45;
            double actual = target.WithdrawMoney(50, currentDate);
            Assert.AreEqual(expected, actual);
        }
```

2. Метод *CheckBalance* класса *CheckingAccount*

```
public void CheckBalanceTest()
        {
            int number = 0;
            string password = string.Empty;
            double balance = 100;
            DateTime openDate = new DateTime(2013,1,1);
            int monthlyQuota = 1;
            double fee = 30;
            CheckingAccount target = new CheckingAccount(number, password, balance, openDate,
monthlyQuota, fee);
            target.PutMoney(10, new DateTime(2013, 1, 4));
            target.PutMoney(5, new DateTime(2013, 1, 9));
            target.WithdrawMoney(5, new DateTime(2013,1,15));
            double expected = 50;
            double actual = target.CheckBalance(new DateTime(2013, 1, 20));
            Assert.AreEqual(expected, actual);
        }
```

3. Метод *CountPenalty* класса *OverdraftAccount*

```
public void CountPenaltyTest()
        {
            int number = 0;
            string password = string.Empty;
            double balance = 10;
            DateTime openDate = new DateTime(2013, 1, 1);
            double int_rate = 5;
            OverdraftAccount target = new OverdraftAccount(number, password, balance, openDate,
int_rate);
            DateTime currDate = new DateTime(2013, 4, 1);
            target.WithdrawMoney(20, new DateTime(2013,2,1));
            double expected = 1.025;
            double actual = target.CountPenalty(currDate);
            Assert.AreEqual(expected, actual);
        }
```

Все тесты были пройдены успешно.

В своем проекте я выбрал Unit-тестирование, поскольку это достаточно простой и эффективный способ проверить правильность работы отдельных небольших кусков кода. Они позволяют легко убедиться в том, что их работа не изменилась после проведения определенных изменений в коде (рефакторинга), соответственно они значительно упрощают обнаружение и устранение ошибок в программе.

Главным недостатком данного вида тестирования является невозможность комплексного тестирования различных блоков программы. Мы можем одновременно проверять работу только определенного блока, поэтому обнаружить ошибки, связанные с интеграцией и производительностью, с их помощью будет невозможно.

Однако текущий проект сложно отнести к категории сложных и крупных проектов, тестирование производительности здесь совершенно необязательно. Поэтому в нем вполне допустимо ограничиться Unit-тестированием, не прибегая к дополнительным методам, так как здесь нам важнее убедиться в правильности функционирования отдельных методов различных классов банковских аккаунтов, ведь реализация остальной части системы, по сути, является тривиальной.