**Национальный исследовательский технологический университет «МИСИС»**

Отчет по лабораторной работе № 6 «Тестирование ПО»

Выполнил:

Миронов Е. А. БИВТ-23-8

Преподаватель:

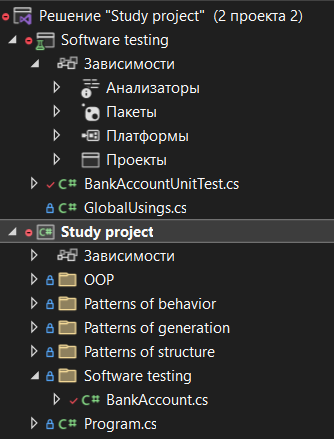
Гласов А. В.

Москва 2024 г.

GitHub repo: <https://github.com/Valet-V0ult-de-Fur1e/programming_technologies_repo>

1. Обновление структуры проекта

В solution создаётся отдельно консольное приложение и отдельно приложение для тестирования классов



Код тестируемого приложения

public class Transaction

{

public float Amount { get; set; }

public DateTime Date { get; set; }

public string Type { get; set; }

public Transaction(float amount, string type)

{

if (amount <= 0)

{

throw new Exception("Amount must be greater than zero.");

}

Amount = amount;

Type = type;

Date = DateTime.Now;

}

}

public interface TransactionsUse

{

public List<Transaction> GetTransactions();

}

public class BankAccount: TransactionsUse

{

private static int accountsCount = 0;

public int Id { get; private set; }

private List<Transaction> \_transactions;

public BankAccount()

{

this.Id = ++accountsCount;

this.\_transactions = new List<Transaction>();

}

public void Deposit(float amount)

{

this.\_transactions.Add(new Transaction(amount, "Deposit"));

}

public void Withdraw(float amount)

{

if (amount > this.getBalance())

{

throw new Exception("Insufficient funds.");

}

this.\_transactions.Add(new Transaction(amount, "Withdraw"));

}

public float getBalance()

{

return \_transactions.Where(t => t.Type == "Deposit").Sum(t => t.Amount) -

\_transactions.Where(t => t.Type == "Withdraw").Sum(t => t.Amount);

}

public List<Transaction> GetTransactions()

{

return this.\_transactions;

}

}

Код Unit тестов

using Moq;

using NUnit.Framework;

using System;

using System.Collections.Generic;

using System.Linq;

[TestFixture]

public class BankAccountTests

{

[Test]

public void DepositValidAmount\_IncreaseBalance()

{

// Arrange

var account = new BankAccount();

var depositAmount = 100f;

// Act

account.Deposit(depositAmount);

// Assert

Assert.That(account.getBalance(), Is.EqualTo(depositAmount));

}

[Test]

public void WithdrawValidAmount\_DecreaseBalance()

{

// Arrange

var account = new BankAccount();

account.Deposit(100f);

var withdrawAmount = 50f;

// Act

account.Withdraw(withdrawAmount);

// Assert

Assert.That(account.getBalance(), Is.EqualTo(50f));

}

[Test]

public void WithdrawInsufficientFunds\_ThrowException()

{

// Arrange

var account = new BankAccount();

account.Deposit(100f);

var withdrawAmount = 150f;

// Act & Assert

var ex = Assert.Throws<Exception>(() => account.Withdraw(withdrawAmount));

Assert.That(ex.Message, Is.EqualTo("Insufficient funds."));

}

[Test]

public void WithdrawInvalidAmount\_ThrowException()

{

// Arrange

var account = new BankAccount();

account.Deposit(100f);

// Act & Assert

var ex = Assert.Throws<Exception>(() => account.Withdraw(-50f));

Assert.That(ex.Message, Is.EqualTo("Amount must be greater than zero."));

}

[Test]

public void WithdrawZeroAmount\_ThrowException()

{

// Arrange

var account = new BankAccount();

account.Deposit(100f);

// Act & Assert

var ex = Assert.Throws<Exception>(() => account.Withdraw(0f));

Assert.That(ex.Message, Is.EqualTo("Amount must be greater than zero."));

}

[Test]

public void GetBalance\_ReturnCorrectBalance()

{

// Arrange

var account = new BankAccount();

account.Deposit(10000f);

account.Withdraw(50f);

// Act

var balance = account.getBalance();

// Assert

Assert.That(balance, Is.EqualTo(9950f));

}

[Test]

public void GetTransactions\_ReturnExpectedTransactions()

{

// Arrange

var mockAccount = new Mock<TransactionsUse>() ;

var mockTransactions = new List<Transaction>

{

new Transaction(100f, "Deposit"),

new Transaction(50f, "Withdraw")

};

mockAccount.Setup(m => m.GetTransactions()).Returns(mockTransactions);

// Act

var transactions = mockAccount.Object.GetTransactions();

// Assert

Assert.That(transactions.Count, Is.EqualTo(2));

Assert.That(transactions.First().Type, Is.EqualTo("Deposit"));

Assert.That(transactions.Last().Type, Is.EqualTo("Withdraw"));

}

}

