МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Надёжность программного обеспечения (НПО)

ОТЧЕТ

по лабораторной работе №7

Тема работы: Юнит тестирование с помощью библиотеки jMock

Выполнил

студент: гр. 551004 Ермошин М. А.

Проверил: Деменковец Д. В.

Минск, 2017

Цель работы - изучение библиотеки jMock и написание юнит тестов для сторонней библиотеки.

**Библиотека для тестирования**

Для тестирования была написана простенькая библиотека на языке java. В ней есть сервис, который умеет получать email из data layer с определённым интерфейсом.

**Тесты**

Была протестирована функция получения email несколькими способами, предоставляющимися в библиотеке jMock:

1. один и только один вызов метода;
2. как минимум 5 вызовов метода;
3. как максимум 5 вызовов метода;
4. вызов метода 0 раз;
5. вызов метода ровно 5 раз;
6. вызов метода, приводящий к исключительной ситуации;

**Текст тестов**

public class ServiceClassTest {  
 private final int ID = 5;  
 private final int WRONG\_ID = 6;  
 private ServiceClass systemUnderTest;  
 private DataAccess mockedDependency;  
 private Mockery mockingContext = new JUnit4Mockery();  
 @Before  
 public void doBeforeEachTestCase() {  
 mockingContext = new JUnit4Mockery();  
 systemUnderTest = new ServiceClassImpl();  
 mockedDependency = mockingContext.mock(DataAccess.class);  
 systemUnderTest.setDataAccess(mockedDependency);  
 }  
 @Test  
 public void getEmail() throws ServiceException {  
 mockingContext.checking(new Expectations() {  
 {  
 oneOf(mockedDependency).getEmail(ID);  
 will(returnValue("myCoolEmail"));  
 }  
 });  
 String testEmail = systemUnderTest.getEmail(ID);  
 }  
  
 @Test(expected = ServiceException.class)  
 public void getEmailNonExistentIdThrowsException() throws ServiceException {  
 mockingContext.checking(new Expectations() {  
 {  
 oneOf(mockedDependency).getEmail(WRONG\_ID);  
 will(returnValue(null));  
 }  
 });  
 final String testEmail = systemUnderTest.getEmail(WRONG\_ID);  
 }  
  
 @Test  
 public void getEmailFiveTimes() throws ServiceException {  
 mockingContext.checking(new Expectations() {  
 {  
 exactly(5).of(mockedDependency).getEmail(ID);  
 will(returnValue("myCoolEmail"));  
 }  
 });  
 String testEmail;  
 for (int i = 0; i<5;i++)  
 testEmail = systemUnderTest.getEmail(ID);  
 }  
  
 @Test  
 public void getEmailZeroTimes() throws ServiceException {  
 mockingContext.checking(new Expectations() {  
 {  
  
 never(mockedDependency).getEmail(ID);  
 will(returnValue("myCoolEmail"));  
 }  
 });  
 String testEmail;  
 for (int i = 0; i<0;i++)  
 testEmail = systemUnderTest.getEmail(ID);  
 }  
  
 @Test  
 public void getEmailAtLeastFiveTimes() throws ServiceException {  
 mockingContext.checking(new Expectations() {  
 {  
 atLeast(5).of(mockedDependency).getEmail(ID);  
 will(returnValue("myCoolEmail"));  
 }  
 });  
 String testEmail;  
 for (int i = 0; i<6;i++)  
 testEmail = systemUnderTest.getEmail(ID);  
 }  
  
 @Test  
 public void getEmailAtMostFiveTimes() throws ServiceException {  
 mockingContext.checking(new Expectations() {  
 {  
 atMost(5).of(mockedDependency).getEmail(ID);  
 will(returnValue("myCoolEmail"));  
 }  
 });  
 String testEmail;  
 for (int i = 0; i<5;i++)  
 testEmail = systemUnderTest.getEmail(ID);  
 }  
}

**Вывод**

В данной лабораторной работы были составлены тесты для модуля my.library библиотеки myLib, а также проверена корректность работы данной библиотеки.