МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федерально автономное бюджетное образовательное учреждение высшего образования

«Севастопольский государственный университет»

кафедра Информационных систем

Куркчи Ариф Эрнестович

Институт информационных технологий и управления в технических системах

курс 3 группа ИС/б-31-о

09.03.02 Информационные системы и технологии (уровень бакалавриата)

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Тестирование программного обеспечения»

на тему «Исследование способов модульного тестирования программного обеспечения в среде JUnit»

Отметка о зачете \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

(дата)

Руководитель практикума

ст. преподаватель   В. А. Строганов

(должность) (подпись) (инициалы, фамилия)

Севастополь 2016

1. ЦЕЛЬ РАБОТЫ

Исследовать эффективность использования методологии TDD при разработке программного обеспечения. Получить практические навыки использования фреймворка JUnit для модульного тестирования программного обеспечения.

1. ПОСТАНОВКА ЗАДАЧИ

Класс-«заглушка» PseudoConsumer, реализующий интерфейс IConsumer. Он должен реализовать работу с его балансом и внутренним хранилищем товаров.

1. ХОД РАБОТЫ
2. Исходный код интерфейса IConsumer

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | package ru.justnero.study.sevsu.pptc; | | 2 |  | | 3 | import *java.util.Map*; | | 4 |  | | 5 | public interface IConsumer { | | 6 |  | | 7 | *double* money(); | | 8 |  | | 9 | *void* money(*double* *amount*, *boolean* *give*); | | 10 |  | | 11 | *Map<Integer, Item>* items(); | | 12 |  | | 13 | *void* item(*Item* *item*, *boolean* *give*); | | 14 |  | | 15 | *void* items(*Map<Integer, Item>* *items*, *boolean* *give*); | | 16 |  | | 17 | } | |  |

1. Исходный код класса PseudoConsumer

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | package ru.justnero.study.sevsu.pptc; | | 2 |  | | 3 | import *sun.reflect.generics.reflectiveObjects.NotImplementedException*; | | 4 |  | | 5 | import *java.lang.reflect.Field*; | | 6 | import *java.util.Map*; | | 7 |  | | 8 | public class PseudoConsumer implements *IConsumer* { | | 9 |  | | 10 | private *String* name; | | 11 | private *double* money; | | 12 | private *Stock* stock; | | 13 |  | | 14 | public PseudoConsumer(*String* *name*, *double* *money*) { | | 15 | this.name = name; | | 16 | this.money = money; | | 17 | stock = *Stock*.getInstance(name+" consumer"); | | 18 | } | | 19 |  | | 20 | public *double* money() { | | 21 | return money; | | 22 | } | | 23 |  | | 24 | *@Override* | | 25 | public *void* money(*double* *amount*, *boolean* *give*) { | | 26 | if(give) { | | 27 | money += amount; | | 28 | } else { | | 29 | if(money >= amount) { | | 30 | money -= amount; | | 31 | } else { | | 32 | throw new *RuntimeException*("Not enough money"); | | 33 | } | | 34 | } | | 35 | } | | 36 |  | | 37 | *@Override* | | 38 | public *Map<Integer, Item>* items() { | | 39 | *Map<Integer, Item>* ret = null; | | 40 | try { | | 41 | *Field* itemsField = *Stock*.class.getDeclaredField("items"); | | 42 | itemsField.setAccessible(true); | | 43 | ret = (*Map<Integer, Item>*) itemsField.get(stock); | | 44 | } catch (*Exception* e) { | | 45 | e.printStackTrace(); | | 46 | } | | 47 | return ret; | | 48 | } | | 49 |  | | 50 | *@Override* | | 51 | public *void* item(*Item* *item*, *boolean* *give*) { | | 52 | if(give) { | | 53 | stock.addItem(item); | | 54 | } else { | | 55 | if(stock.takeItem(item.getId(), item.getQuantity()) == null) { | | 56 | throw new *RuntimeException*("No such item, or not enough"); | | 57 | } | | 58 | } | | 59 | } | | 60 |  | | 61 | *@Override* | | 62 | public *void* items(*Map<Integer, Item>* *items*, *boolean* *give*) { | | 63 | for(*Item* item : items.values()) { | | 64 | if(give) { | | 65 | stock.addItem(item); | | 66 | } else { | | 67 | if(stock.takeItem(item.getId(), item.getQuantity()) == null) { | | 68 | throw new *RuntimeException*("No such item, or not enough"); | | 69 | } | | 70 | } | | 71 | } | | 72 | } | | 73 | } | |  |

1. Исходный код теста PseudoConsumerTest

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | package ru.justnero.study.sevsu.pptc; | | 2 |  | | 3 | import *org.junit.Before*; | | 4 | import *org.junit.Test*; | | 5 |  | | 6 | import *java.math.BigInteger*; | | 7 | import *java.util.Random*; | | 8 |  | | 9 | import static org.hamcrest.CoreMatchers.\*; | | 10 | import static org.junit.Assert.\*; | | 11 |  | | 12 | public class PseudoConsumerTest { | | 13 |  | | 14 | private *IConsumer* consumer; | | 15 |  | | 16 | *@Before* | | 17 | public *void* setUp() { | | 18 | *String* name  = new *BigInteger*(130, new *Random*()).toString(32); | | 19 | consumer = new *PseudoConsumer*(name, 15.3D); | | 20 | consumer.item(new *Item*(2, "Test 2", 43.21D, 4), true); | | 21 | } | | 22 |  | | 23 | *@Test* | | 24 | public *void* constructTest() { | | 25 | assertThat(consumer.money(), is(15.3D)); | | 26 | assertThat(consumer.items().size(), is(1)); | | 27 | } | | 28 |  | | 29 | *@Test* | | 30 | public *void* itemGiveTest() { | | 31 | consumer.item(new *Item*(1, "Test 1", 12.34D, 1), true); | | 32 | consumer.item(new *Item*(1, "Test 1", 12.34D, 2), true); | | 33 |  | | 34 | assertThat(consumer.items().size(), is(2)); | | 35 | assertThat(consumer.items().get(1).getQuantity(), is(3)); | | 36 | assertThat(consumer.items().get(2).getQuantity(), is(4)); | | 37 | } | | 38 |  | | 39 | *@Test* | | 40 | public *void* itemTakeTest() { | | 41 | consumer.item(new *Item*(2, "Test 2", 43.21D, 1), false); | | 42 |  | | 43 | assertThat(consumer.items().get(2).getQuantity(), is(3)); | | 44 | } | | 45 |  | | 46 | *@Test* | | 47 | public *void* moneyGiveTest() { | | 48 | consumer.money(12.34D, true); | | 49 |  | | 50 | assertThat(consumer.money(), is(15.3D + 12.34D)); | | 51 | } | | 52 |  | | 53 | *@Test* | | 54 | public *void* moneyTakeTest() { | | 55 | try { | | 56 | consumer.money(43.21D, false); | | 57 | } catch (*RuntimeException* ex) { | | 58 | assertThat(ex.getMessage(), equalTo("Not enough money")); | | 59 | } | | 60 |  | | 61 | consumer.money(12.34D, false); | | 62 | assertThat(consumer.money(), is(15.3D - 12.34D)); | | 63 |  | | 64 | } | | 65 |  | | 66 | } | |  |

Тестовые случаи рассматривают:

1. Создание нового экземпляра класса PseudoConsumerTest с проверкой его начального баланса и предметов
2. Выдачу товаров и проверку их наличия и группировки в хранилище покупателя
3. Изъятие товара из хранилища покупателя с проверкой количества остатка в хранилище
4. Добавление денег на баланс пользователя с проверкой итогового остатка
5. Снятие денег с баланса пользователя с проверкой при попытке снятия большей суммы, чем имеется на остатке в данный момент

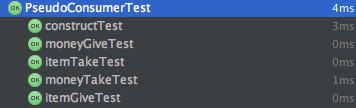


Рисунок 1 – Результат выполнения тестов

ВЫВОДЫ

В ходе данной лабораторной работы были изучены подходы к разработке ПО с использованием методологии TDD.