ИАТЭ НИЯУ МИФИ

Кафедра Компьютерных систем, сетей и технологий

Отчёт по лабораторной работе №3 по ТПРГ

на тему:

«Тестирование и рефакторинг»

Выполнили: студенты гр. ВТ-Б12

Кругликов А. Д.

Пастушак Р. Л.

Донцов М. А.

Проверил: Тельнов В. П.

Обнинск 2014г.

Содержание:

1. Постановка задачи.
2. Сценарии тестирования, тестовые условия и наборы данных.
   1. Для модульного тестирования
   2. Для интеграционного тестирования
   3. Для системного тестирования
3. Листинг кода разработанных драйверов тестирования.
4. Протоколы тестирования в табличной форме. Выявленные и устранённые ошибки.
5. Результаты тестирования, рефакторинга и оптимизации кода, вывод по работе.

1.Постановка задачи.

Провести оптимизацию, рефакторинг, тестирование продукта, разработанного в лабораторных работах №1 и №2.  
Тестирование провести в режиме «чёрного ящика» и «белого ящика» следующими видами :

1. Модульное тестирование.
2. Интеграционное тестирование.
3. Системное тестирование.

2. Сценарии тестирования, тестовые условия и наборы данных.

a) Модульное тестирование.

Сценарий тестирования в режиме «чёрного ящика»:

* Класс Distribution

- Метод GetFloatCount

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Тесты | 0 | Big | BigVersa | Drobnoe | MinMinus | NoNorm | Norm | MinusDrobnoe | MinusMinus | Munus1 |
| Переменные | float count | 0 | 9999999999999999999999999999999 | 1 | 3.23456 | -11 | 6 | 24 | -5.4321 | -532 | -1 |
| float min | 0 | 1 | 9999999999999999999999999999 | 1.23456 | 0 | 24 | 6 | -10.9876543 | -35 | -1 |
| Тестовые условия | result >= 0  result >= min | | | | | | | result >= min | | |

- Метод GetIntCount

Тестовые условия:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Тесты | 0 | BigNorm | Drobnoe | Minus1 | MinusDrobnoe | Norm |
| Переменные | float count | 0 | 999999999999999999999999 | 1.5 | -1 | -14.2 | 33 |
| Тестовые условия | result >= 0 | | | | | |

- Метод GetBuyers

Входные данные:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тесты | 0 | 99999 | 9999999999 | Random | Minus1 |
| Переменные | int time | 0 | 99999 | 9999999999 | ? | -1 |
| int clients | 0 | 99999 | 9999999999 | ? | -1 |
| double[] IdealSum | 0 | 99999 | 999999999999999999 | ? | -1 |
| int[] RealSum | 0 | 99999 | 9999999999 | ? | -1 |
| Тестовые условия | IdealSum < RealSum  RealSum==0 | | | | IdealSum < RealSum, RealSum==-1 |

* Класс GeneratorTest

- Метод CreateCart

Входные данные:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Тесты | BigCount | NegativeCart | NegativeValue | Not0 | Null | TooBig |
| Переменные | float[] cart | {0} | {-9, -9, -9, -9, -9, -9} | {0} | {9999, 99999, 9, 9, 9, 9} | null | {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} |
| int count | 9999999 | 5 | -1 | 5 | -1 | 3 |
| Тестовые условия | Метод не выдаёт ошибок | | | | | |

* Класс SQLAgentTest

- Метод AddProduct

Входные данные:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Тесты | Dublicate | NegativeCode | NegativeCount | NegativePrice | Normal | NullName | Null |
| Переменные | Product item | - | - | - | - | - | - | null |
| item.setCode | 999 | -666999 | 1000…2000 | 1000…2000 | 1000…2000 | 1000…2000 | - |
| item.setCount | 3.4 | 3.4 | -3.4 | 3.4 | 3.4 | 3.4 | - |
| item.setIsPacked | false | false | false | false | false | false | - |
| item.setName | "Abacaba" | "Abacaba" | "Abacaba" | "Abacaba" | "Abacaba" | null | - |
| item.setPrice | 0…1000 | 0…1000 | 0…1000 | -1000…0 | 0…1000 | 0…1000 | - |
| Тестовые условия | result = false | | | | Result=true | result = false | |

- Метод Buyed

Входные данные:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Тесты | MixCart | NegCart | Negbuyerid | Negsimid | Negtime | Normal |
| Переменные | int buyerid | 0…1000 | 0…1000 | -1000…0 | 0…1000 | 0…1000 | 0…1000 |
| int time | 0…2880 | 0…2880 | 0…2880 | 0…2880 | -2880…0 | 0…2880 |
| int simid | 50 | 200…300 | 200…300 | -50 | 50 | 50 |
| float[] cart | {1, -2, 3, -4, 5, -6, 7, -8, 9} | {-1, -2, -3, -4, -5, -6, -7, -8, -9} | {1, 2, 3, 4, 5, 6, 7, 8, 9} | {1, 2, 3, 4, 5, 6, 7, 8, 9} | {1, 2, 3, 4, 5, 6, 7, 8, 9} | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| Тестовые условия | result >= 0 | | | | | |

- Метод Connect

Входные данные:

|  |  |
| --- | --- |
| Тестовые условия | Метод не выдаёт ошибок |

- Метод DeleteProduct

Входные данные:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тесты | Invalid Code | Negative Code | Null Code | Normal |
| Переменные | int code | Не существующий | -50 | null | Существующий |
| Тестовые условия | Result = false | | | Result = true |

- Метод Disconnect

Входные данные:

|  |  |  |
| --- | --- | --- |
| Тесты | Disconnect | DisconnectConnected |
| Тестовые условия | Метод не выдаёт ошибок | |

- Метод Ended

Входные данные:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Тесты | 0 | 123 | 999999 | Disconnected | Negative | NoSimId |
| Переменные | int peoplearrived | 0 | 123 | 999999 | 50 | -50 | 50 |
| int peopleserved | 0 | 123 | 999999 | 50 | -50 | 50 |
| float avggoodscount | 0 | 123 | 999999.999999 | 50 | -50 | 50 |
| float avgprofit | 0 | 123 | 999999. 999999 | 50 | -50 | 50 |
| int profit | 0 | 123 | 999999 | 50 | -50 | 50 |
| int maxqueue | 0 | 123 | 999999 | 50 | -50 | 50 |
| int maxqueuetime | 0 | 123 | 999999 | 50 | -50 | 50 |
| int simid | Существующий | Существующий | Существующий | Существующий | Существующий | Не существующий |
| Тестовые условия | Метод не выдаёт ошибок | | | | | |

- Метод Fix

Входные данные:

|  |  |  |
| --- | --- | --- |
| Тесты | Connected | Disconnected |
| Тестовые условия | Метод не выдаёт ошибок | |

- Метод GetResults

Входные данные:

|  |  |  |  |
| --- | --- | --- | --- |
| Тесты | Invalid | NegId | Valid |
| Переменные | int simulationid | Не существующий | -50 | Существующий |
| Тестовые условия | Должна быть на выходе одна строка | | Должно быть на выходе больше одной строки |

- Метод Load

Входные данные:

|  |  |  |
| --- | --- | --- |
| Тесты | Connected | Disconnected |
| Тестовые условия | Размер списка больше 0 | Размер списка равен 0 |

- Метод LoadSettings

Входные данные:

|  |  |  |  |
| --- | --- | --- | --- |
| Тесты | Invalid | Nofile | Valid |
| Тестовые условия | result=false | | result=true |

- Метод ShowProductInfo

Входные данные:

|  |  |
| --- | --- |
| Тестовые условия | Метод не выдаёт ошибок |

- Метод Started

Входные данные:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тесты | 0 | 999999 | Negative | Normal | NormalDisconnected |
| Переменные | int peoplecount | 0 | 999999 | -100 | 100 | 100 |
| int goodscount | 0 | 999999 | -5 | 5 | 5 |
| Тестовые условия | result > 0 | | result == 0 | result > 0 | result == 0 |

- Метод TestConnect

Входные данные:

|  |  |  |
| --- | --- | --- |
| Тесты | Connected | Disconnected |
| Тестовые условия | Подключение к базе есть | Подключение к базе нет |

- Метод UpdateProduct

Входные данные:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Тесты | EmptyName | InvalidCode | NegCode | NegCount | NegPrice | Normal | NullName | NullProd |
| Переменные | item.setCode | Существующий | Не существующий | Отрицательный код | Существующий | Существующий | Существующий | Существующий | null |
| item.setCount | 3.4 | 3.4 | 3.4 | -3.4 | 3.4 | 3.4 | 3.4 | null |
| item.setIsPacked | false | false | false | false | false | false | false | null |
| item.setName | "" | "Abacaba" | "Abacaba" | "Abacaba" | "Abacaba" | "Abacaba" | null | null |
| item.setPrice | 0…1000 | 0…1000 | 0…1000 | 0…1000 | …<0 | 0…1000 | 0…1000 | null |
| Тестовые условия | Товар не изменён | | | | | Товар изменён | Товар не изменён | |

Сценарий тестирования в режиме «белого ящика»:

Было проведено тестирование режиме «белого ящика» для тех же методов, что и в режиме «чёрного ящика» при тех же входных наборах данных. В добавок к этому было проведено тестирование ещё пяти методов других классов.

* Класс DistributionTest

- Метод GetBuyers

Входные данные:

|  |  |  |
| --- | --- | --- |
| Тесты | \_ | Minus1 |
| Переменные | int time | 5 | -1 |
| int clients | 100 | -1 |
| double[] IdealSum | {1.9999999} | -1 |
| int[] RealSum | {0} | -1 |
| Тестовые условия | IdealSum[0] >= RealSum[0]  result >= 0  RealSum[0] == 0 | result >= 0 |

- Метод GetFloatCount

Входные данные:

|  |  |
| --- | --- |
| Тесты | \_ |
| Переменные | float count | 5 |
| float min | 0.5 |
| Тестовые условия | result >= 0  result >= min |

- Метод GetIntCount

Входные данные:

|  |  |
| --- | --- |
| Тесты | \_ |
| Переменные | float count | 5 |
| Тестовые условия | result >= 0 |

* Класс WhiteGeneratorTest

- Метод CreateCart

Входные данные:

|  |  |
| --- | --- |
| Тесты | \_ |
| Переменные | float[] cart | {0, 0, 0, 0, 0, 0, 0, 0} |
| int count | 4 |
| Тестовые условия | Кол-во каждого товара в корзине > 0 |

* WhiteWorkerTest

- Метод StartSim

|  |  |
| --- | --- |
| Тесты | \_ |
| Переменные | int peoplecount | 100 |
| int goodscount | 4 |
| Тестовые условия | Метод не выдаёт ошибок |

b) Интеграционное тестирование.

Интеграционное тестирование проверяет связь между классами и методами этих классов. В режиме тестирования «белого ящика» было получено покрытие практически всего исходного кода программы, в ходе чего были проверены связи между методами этих классов.  
В связях между методами различных классов ошибок не было выявлено.

с) Системное тестирование.

Общее тестирование программы;  
Тестирование добавления товаров в базу;  
Тестирование изменения товаров в базе;  
Тестирование удаления товаров в базе;

* подавались различные входные данные; проверялся отклик программы на некорректные данные. При корректных данных проверялись отчёты о симуляциях и списки товаров в базе.

Измерялось время проведения симуляции с точки зрения пользователя.

## 3. Листинг кода разработанных драйверов тестирования. Драйверы тестирования

Все драйверы находятся в пакете org.aba.tradingterminal

### DistributionTest

import org.junit.Test;

import static org.junit.Assert.\*;

public class DistributionTest {

public DistributionTest() {

}

@Test

public void testGetIntCount0() {

System.out.println("GetIntCount");

float count = 0;

int result = 0;

try {

result = Distribution.GetIntCount(count);

assertTrue(result >= 0);

} catch (Exception ex) {

fail("GetIntCount - exception! count=" + count + " result=" + result);

}

}

@Test

public void testGetIntCountMinus1() {

System.out.println("GetIntCount");

float count = -1;

int result = 0;

try {

result = Distribution.GetIntCount(count);

assertTrue(result >= 0);

} catch (Exception ex) {

fail("GetIntCount - exception! count=" + count + " result=" + result);

}

}

@Test

public void testGetIntCountDrobnoe() {

System.out.println("GetIntCount");

float count = (float) 1.5;

int result = 0;

try {

result = Distribution.GetIntCount(count);

assertTrue(result >= 0);

} catch (Exception ex) {

fail("GetIntCount - exception! count=" + count + " result=" + result);

}

}

@Test

public void testGetIntCountMinusDrobnoe() {

System.out.println("GetIntCount");

float count = (float) -14.2;

int result = 0;

try {

result = Distribution.GetIntCount(count);

assertTrue(result >= 0);

} catch (Exception ex) {

fail("GetIntCount - exception! count=" + count + " result=" + result);

}

}

@Test

public void testGetIntCountNorm() {

System.out.println("GetIntCount");

float count = 33;

int result = 0;

try {

result = Distribution.GetIntCount(count);

assertTrue(result >= 0);

} catch (Exception ex) {

fail("GetIntCount - exception! count=" + count + " result=" + result);

}

}

@Test

public void testGetIntCountBigNorm() {

System.out.println("GetIntCount");

float count = 999999999999999999999999f;

int result = 0;

try {

result = Distribution.GetIntCount(count);

assertTrue(result >= 0);

} catch (Exception ex) {

fail("GetIntCount - exception! count=" + count + " result=" + result);

}

}

@Test

public void testGetFloatCount0() {

System.out.println("GetFloatCount");

float count = 0;

float min = 0;

float result = 0;

try {

result = Distribution.GetFloatCount(count, min);

assertTrue(result >= 0);

assertTrue(result >= min);

} catch (Exception ex) {

fail("GetFloatCount - exception! count=" + count + " min=" + min + "result=" + result);

}

}

@Test

public void testGetFloatCountMinMinus() {

System.out.println("GetFloatCount");

float count = -11;

float min = 0;

float result = 0;

try {

result = Distribution.GetFloatCount(count, min);

assertTrue(result >= 0);

assertTrue(result >= min);

} catch (Exception ex) {

fail("GetFloatCount - exception! count=" + count + " min=" + min + "result=" + result);

}

}

@Test

public void testGetFloatCountNorm() {

System.out.println("GetFloatCount");

float count = 24;

float min = 6;

float result = 0;

try {

result = Distribution.GetFloatCount(count, min);

assertTrue(result >= 0);

assertTrue(result >= min);

} catch (Exception ex) {

fail("GetFloatCount - exception! count=" + count + " min=" + min + "result=" + result);

}

}

@Test

public void testGetFloatCountNoNorm() {

System.out.println("GetFloatCount");

float count = 6;

float min = 24;

float result = 0;

try {

result = Distribution.GetFloatCount(count, min);

assertTrue(result >= 0);

assertTrue(result >= min);

} catch (Exception ex) {

fail("GetFloatCount - exception! count=" + count + " min=" + min + "result=" + result);

}

}

@Test

public void testGetFloatCountMinusMinus() {

System.out.println("GetFloatCount");

float count = -532;

float min = -35;

float result = 0;

try {

result = Distribution.GetFloatCount(count, min);

assertTrue(result >= min);

} catch (Exception ex) {

fail("GetFloatCount - exception! count=" + count + " min=" + min + "result=" + result);

}

}

@Test

public void testGetFloatCountBig() {

System.out.println("GetFloatCount");

float count = 9999999999999999999999999999999f;

float min = 1;

float result = 0;

try {

result = Distribution.GetFloatCount(count, min);

assertTrue(result >= 0);

assertTrue(result >= min);

} catch (Exception ex) {

fail("GetFloatCount - exception! count=" + count + " min=" + min + "result=" + result);

}

}

@Test

public void testGetFloatCountBigVersa() {

System.out.println("GetFloatCount");

float count = 1;

float min = 9999999999999999999999999999f;

float result = 0;

try {

result = Distribution.GetFloatCount(count, min);

assertTrue(result >= 0);

assertTrue(result >= min);

} catch (Exception ex) {

fail("GetFloatCount - exception! count=" + count + " min=" + min + "result=" + result);

}

}

@Test

public void testGetFloatCountMunus1() {

System.out.println("GetFloatCount");

float count = -1;

float min = -1;

float result = 0;

try {

result = Distribution.GetFloatCount(count, min);

assertTrue(result >= min);

} catch (Exception ex) {

fail("GetFloatCount - exception! count=" + count + " min=" + min + "result=" + result);

}

}

@Test

public void testGetFloatCountDrobnoe() {

System.out.println("GetFloatCount");

float count = (float) 3.23456;

float min = (float) 1.23456;

float result = 0;

try {

result = Distribution.GetFloatCount(count, min);

assertTrue(result >= 0);

assertTrue(result >= min);

} catch (Exception ex) {

fail("GetFloatCount - exception! count=" + count + " min=" + min + "result=" + result);

}

}

@Test

public void testGetFloatCountMinusDrobnoe() {

System.out.println("GetFloatCount");

float count = (float) -5.4321;

float min = (float) -10.9876543;

float result = 0;

try {

result = Distribution.GetFloatCount(count, min);

assertTrue(result >= min);

} catch (Exception ex) {

fail("GetFloatCount - exception! count=" + count + " min=" + min + "result=" + result);

}

}

@Test

public void testGetBuyers0() {

System.out.println("GetBuyers");

int time = 0;

int clients = 0;

double[] IdealSum = {0};

int[] RealSum = {0};

int result = 0;

try {

result = Distribution.GetBuyers(time, clients, IdealSum, RealSum);

assertTrue("IdealSum < RealSum", IdealSum[0] >= RealSum[0]);

assertEquals("RealSum!=0", 0, RealSum[0]);

} catch (Exception ex) {

fail("GetBuyers - exception! time=" + time + " clients=" + clients + " IdealSum=" + IdealSum[0] + " RealSum=" + RealSum[0] + " result=" + result);

}

}

@Test

public void testGetBuyersMinus1() {

System.out.println("GetBuyers");

int time = -1;

int clients = -1;

double[] IdealSum = {-1};

int[] RealSum = {-1};

int result = 0;

try {

result = Distribution.GetBuyers(time, clients, IdealSum, RealSum);

assertTrue("IdealSum < RealSum", IdealSum[0] >= RealSum[0]);

assertEquals("RealSum!=0", -1, RealSum[0]);

} catch (Exception ex) {

fail("GetBuyers - exception! time=" + time + " clients=" + clients + " IdealSum=" + IdealSum[0] + " RealSum=" + RealSum[0] + " result=" + result);

}

}

@Test

public void testGetBuyersRandom() {

System.out.println("GetBuyers");

int time = (int) Math.random() + 5;

int clients = (int) Math.random() + 5;

double[] IdealSum = {Math.random()};

int[] RealSum = {(int) Math.random()};

int result = 0;

try {

result = Distribution.GetBuyers(time, clients, IdealSum, RealSum);

assertTrue("IdealSum < RealSum", IdealSum[0] >= RealSum[0]);

assertEquals("RealSum!=0", 0, RealSum[0]);

} catch (Exception ex) {

fail("GetBuyers - exception! time=" + time + " clients=" + clients + " IdealSum=" + IdealSum[0] + " RealSum=" + RealSum[0] + " result=" + result);

}

}

@Test

public void testGetBuyers99999() {

System.out.println("GetBuyers");

int time = 99999;

int clients = 99999;

double[] IdealSum = {99999};

int[] RealSum = {99999};

int result = 0;

try {

result = Distribution.GetBuyers(time, clients, IdealSum, RealSum);

assertTrue("IdealSum < RealSum", IdealSum[0] >= RealSum[0]);

assertEquals("RealSum!=0", 0, RealSum[0]);

} catch (Exception ex) {

fail("GetBuyers - exception! time=" + time + " clients=" + clients + " IdealSum=" + IdealSum[0] + " RealSum=" + RealSum[0] + " result=" + result);

}

}

@Test

public void testGetBuyers9999999999() {

System.out.println("GetBuyers");

int time = 999999999;

int clients = 999999999;

double[] IdealSum = {999999999999999999D};

int[] RealSum = {999999999};

int result = 0;

try {

result = Distribution.GetBuyers(time, clients, IdealSum, RealSum);

assertTrue("IdealSum < RealSum", IdealSum[0] >= RealSum[0]);

assertEquals("RealSum!=0", 0, RealSum[0]);

} catch (Exception ex) {

fail("GetBuyers - exception! time=" + time + " clients=" + clients + " IdealSum=" + IdealSum[0] + " RealSum=" + RealSum[0] + " result=" + result);

}

}

}

### GeneratorTest

import org.junit.Test;

import static org.junit.Assert.\*;

public class GeneratorTest {

public GeneratorTest() {

}

@Test

public void testCreateCartNegativeValue() {

System.out.println("CreateCart");

float[] cart = {0};

int count = -1;

boolean negative = false;

try {

Generator.CreateCart(cart, count);

for (int i = 0; i < cart.length; i++) {

if (cart[i] < 0) {

negative = true;

}

}

assertFalse("Some of products have negative value", negative);

} catch (Exception ex) {

fail("Unexpected exeption - negative");

}

}

@Test

public void testCreateCartNull(){

System.out.println("CreateCart");

float[] cart = null;

int count = -1;

Generator.CreateCart(cart, count);

}

@Test

public void testCreateCartTooBig() {

System.out.println("CreateCart");

float[] cart = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

int count = 3;

try {

Generator.CreateCart(cart, count);

} catch (Exception ex) {

fail("Unexpected exeption - too big");

}

}

@Test

public void testCreateCartNot0() {

System.out.println("CreateCart");

float[] cart = {9999, 99999, 9, 9, 9, 9};

int count = 5;

try {

Generator.CreateCart(cart, count);

} catch (Exception ex) {

fail("Unexpected exeption - not 0");

}

}

@Test

public void testCreateCartBigCount() {

System.out.println("CreateCart");

float[] cart = {0};

int count = 9999999;

try {

Generator.CreateCart(cart, count);

} catch (Exception ex) {

fail("Unexpected exeption - big count");

}

}

@Test

public void testCreateCartNegativeCart() {

System.out.println("CreateCart");

float[] cart = {-9, -9, -9, -9, -9, -9};

int count = 5;

try {

Generator.CreateCart(cart, count);

} catch (Exception ex) {

fail("Unexpected exeption - negative cart");

}

}

}

### SQLAgentTest

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.util.LinkedList;

import java.util.Random;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.junit.Test;

import static org.junit.Assert.\*;

import org.junit.Before;

public class SQLAgentTest {

public SQLAgentTest() {

}

@Before

public void setUp() {

try {

FileWriter fw;

fw = new FileWriter("DBProps.prop");

BufferedWriter bw = new BufferedWriter(fw);

bw.append("tradingterminal2\n");

bw.append("50.50.0.106:3306\n"); // Network address!!!

bw.append("kramer98489\n");

bw.append("4321\n");

bw.flush();

bw.close();

fw.flush();

fw.close();

} catch (IOException ex) {

Logger.getLogger(SQLAgentTest.class.getName()).log(Level.SEVERE, null, ex);

}

}

@Test

public void testConnect() {

System.out.println("Connect");

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

} catch (Exception ex) {

fail("Connect error");

}

}

@Test

public void testDisconnect() {

System.out.println("Disconnect");

try {

SQLAgent.Disconnect();

} catch (Exception ex) {

fail("Disconnect error");

}

}

@Test

public void testDisconnectConnected() {

System.out.println("Disconnect");

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Disconnect();

} catch (Exception ex) {

fail("Disconnect connected error");

}

}

@Test

public void testTestConnectDisconnected() {

System.out.println("TestConnect");

boolean expResult = false;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Disconnect();

boolean result = SQLAgent.TestConnect();

assertEquals(expResult, result);

} catch (Exception ex) {

fail("Test connect - disconnected error");

}

}

@Test

public void testTestConnectConnected() {

System.out.println("TestConnect");

boolean expResult = true;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

boolean result = SQLAgent.TestConnect();

assertEquals(expResult, result);

} catch (Exception ex) {

fail("Test connect - connected error");

}

}

@Test

public void testStartedNormal() {

System.out.println("Started");

int peoplecount = 100;

int goodscount = 5;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

int result = SQLAgent.Started(peoplecount, goodscount);

assertTrue("simid <= 0 ???", result > 0);

} catch (Exception ex) {

fail("Starting failed");

}

}

@Test

public void testStartedNormalDisconnected() {

System.out.println("Started");

int peoplecount = 100;

int goodscount = 5;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Disconnect();

int result = SQLAgent.Started(peoplecount, goodscount);

assertTrue("simid != 0 ???", result == 0);

} catch (Exception ex) {

fail("Starting failed");

}

}

@Test

public void testStartedNegative() {

System.out.println("Started");

int peoplecount = -100;

int goodscount = -5;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

int result = SQLAgent.Started(peoplecount, goodscount);

assertTrue("simid != 0 ???", result == 0);

} catch (Exception ex) {

fail("Starting negative failed");

}

}

@Test

public void testStarted0() {

System.out.println("Started");

int peoplecount = 0;

int goodscount = 0;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

int result = SQLAgent.Started(peoplecount, goodscount);

assertTrue("simid <= 0 ???", result > 0);

} catch (Exception ex) {

fail("Starting 0 failed");

}

}

@Test

public void testStarted999999() {

System.out.println("Started");

int peoplecount = 999999;

int goodscount = 999999;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

int result = SQLAgent.Started(peoplecount, goodscount);

assertTrue("simid <= 0 ???", result > 0);

} catch (Exception ex) {

fail("Starting 999999 failed");

}

}

@Test

public void testEnded0() {

System.out.println("Ended");

int peoplearrived = 0;

int peopleserved = 0;

float avggoodscount = 0.0F;

float avgprofit = 0.0F;

int profit = 0;

int maxqueue = 0;

int maxqueuetime = 0;

int simid = 20;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Ended(peoplearrived, peopleserved, avggoodscount, avgprofit, profit, maxqueue, maxqueuetime, simid);

} catch (Exception ex) {

fail("Ended 0");

}

}

@Test

public void testEnded123() {

System.out.println("Ended");

int peoplearrived = 132;

int peopleserved = 123;

float avggoodscount = 123.0F;

float avgprofit = 123.0F;

int profit = 123;

int maxqueue = 123;

int maxqueuetime = 123;

int simid = 20;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Ended(peoplearrived, peopleserved, avggoodscount, avgprofit, profit, maxqueue, maxqueuetime, simid);

} catch (Exception ex) {

fail("Ended 123");

}

}

@Test

public void testEndedNoSimId() {

System.out.println("Ended");

int peoplearrived = 1;

int peopleserved = 1;

float avggoodscount = 1.0F;

float avgprofit = 1.0F;

int profit = 1;

int maxqueue = 1;

int maxqueuetime = 1;

int simid = 999999;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Ended(peoplearrived, peopleserved, avggoodscount, avgprofit, profit, maxqueue, maxqueuetime, simid);

} catch (Exception ex) {

fail("Ended no simid");

}

}

@Test

public void testEndedNegative() {

System.out.println("Ended");

int peoplearrived = -50;

int peopleserved = -50;

float avggoodscount = -50.0F;

float avgprofit = -50.0F;

int profit = -50;

int maxqueue = -50;

int maxqueuetime = -50;

int simid = 20;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Ended(peoplearrived, peopleserved, avggoodscount, avgprofit, profit, maxqueue, maxqueuetime, simid);

} catch (Exception ex) {

fail("The test case is a prototype.");

}

}

@Test

public void testEnded999999() {

System.out.println("Ended");

int peoplearrived = 999999;

int peopleserved = 999999;

float avggoodscount = 999999.999999F;

float avgprofit = 999999.999999F;

int profit = 999999;

int maxqueue = 999999;

int maxqueuetime = 999999;

int simid = 20;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Ended(peoplearrived, peopleserved, avggoodscount, avgprofit, profit, maxqueue, maxqueuetime, simid);

} catch (Exception ex) {

fail("Ended 999999");

}

}

@Test

public void testEndedDisconnected() {

System.out.println("Ended");

int peoplearrived = 12;

int peopleserved = 12;

float avggoodscount = 12.0F;

float avgprofit = 12.0F;

int profit = 12;

int maxqueue = 21;

int maxqueuetime = 2122;

int simid = 20;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Disconnect();

SQLAgent.Ended(peoplearrived, peopleserved, avggoodscount, avgprofit, profit, maxqueue, maxqueuetime, simid);

} catch (Exception ex) {

fail("Ended disconnected");

}

}

@Test

public void testBuyedNormal() {

System.out.println("Buyed");

Random RND = new Random();

int buyerid = RND.nextInt(1000);

int time = RND.nextInt(2880);

int simid = 50;

float[] cart = {1, 2, 3, 4, 5, 6, 7, 8, 9};

boolean discount = false;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Load();

int result = SQLAgent.Buyed(buyerid, time, simid, cart, discount);

assertTrue("Total < 0", result >= 0);

} catch (Exception ex) {

fail("Buyed Normal faied");

}

}

@Test

public void testBuyedNegbuyerid() {

System.out.println("Buyed");

Random RND = new Random();

int buyerid = -RND.nextInt(1000);

int time = RND.nextInt(2880);

int simid = 200 + RND.nextInt(100);

float[] cart = {1, 2, 3, 4, 5, 6, 7, 8, 9};

boolean discount = false;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Load();

int result = SQLAgent.Buyed(buyerid, time, simid, cart, discount);

assertTrue("Total < 0", result >= 0);

} catch (Exception ex) {

fail("Buyed Negbuyerid faied" + ex);

}

}

@Test

public void testBuyedNegtime() {

System.out.println("Buyed");

Random RND = new Random();

int buyerid = RND.nextInt(1000);

int time = -RND.nextInt(2880);

int simid = 50;

float[] cart = {1, 2, 3, 4, 5, 6, 7, 8, 9};

boolean discount = false;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Load();

int result = SQLAgent.Buyed(buyerid, time, simid, cart, discount);

assertTrue("Total < 0", result >= 0);

} catch (Exception ex) {

fail("Buyed Neg time faied" + ex);

}

}

@Test

public void testBuyedNegsimid() {

System.out.println("Buyed");

Random RND = new Random();

int buyerid = RND.nextInt(1000);

int time = RND.nextInt(2880);

int simid = -50;

float[] cart = {1, 2, 3, 4, 5, 6, 7, 8, 9};

boolean discount = false;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Load();

int result = SQLAgent.Buyed(buyerid, time, simid, cart, discount);

assertTrue("Total < 0", result >= 0);

} catch (Exception ex) {

fail("Buyed Neg simid faied" + ex);

}

}

@Test

public void testBuyedNegCart() {

System.out.println("Buyed");

Random RND = new Random();

int buyerid = RND.nextInt(1000);

int time = RND.nextInt(2880);

int simid = 200 + RND.nextInt(100);

float[] cart = {-1, -2, -3, -4, -5, -6, -7, -8, -9};

boolean discount = false;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Load();

int result = SQLAgent.Buyed(buyerid, time, simid, cart, discount);

assertTrue("Total < 0", result >= 0);

} catch (Exception ex) {

fail("Buyed Neg cart faied" + ex);

}

}

@Test

public void testBuyedMixCart() {

System.out.println("Buyed");

Random RND = new Random();

int buyerid = RND.nextInt(1000);

int time = RND.nextInt(2880);

int simid = 50;

float[] cart = {1, -2, 3, -4, 5, -6, 7, -8, 9};

boolean discount = false;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Load();

int result = SQLAgent.Buyed(buyerid, time, simid, cart, discount);

assertTrue("Total < 0", result >= 0);

} catch (Exception ex) {

fail("Buyed Mix cart faied" + ex);

}

}

@Test

public void testShowProductInfo() {

System.out.println("ShowProductInfo");

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Load();

LinkedList<String> result = SQLAgent.ShowProductInfo();

} catch (Exception ex) {

fail("ShowProductInfo Failed");

}

}

@Test

public void testLoadConnected() {

System.out.println("Load");

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Load();

assertTrue("Not loaded", SQLAgent.getRangeofGoods().size() > 0);

} catch (Exception ex) {

fail("Load error");

}

}

@Test

public void testLoadDisconnected() {

System.out.println("Load");

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Disconnect();

SQLAgent.Load();

assertTrue("Not loaded", SQLAgent.getRangeofGoods().size() == 0);

} catch (Exception ex) {

fail("Load disconnected error");

}

}

@Test

public void testGetResultsNegId() {

System.out.println("GetResults");

int simulationid = -50;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Load();

LinkedList<String> result = SQLAgent.GetResults(simulationid);

assertFalse("More than one string", result.size() > 1);

} catch (Exception ex) {

fail("Neg Id Failed " + ex);

}

}

@Test

public void testGetResultsInvalid() {

System.out.println("GetResults");

int simulationid = 1234568;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Load();

LinkedList<String> result = SQLAgent.GetResults(simulationid);

assertFalse("More than one string", result.size() > 1);

} catch (Exception ex) {

fail("Invalid Id Failed " + ex);

}

}

@Test

public void testGetResultsValid() {

System.out.println("GetResults");

int simulationid = 5;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Load();

LinkedList<String> result = SQLAgent.GetResults(simulationid);

assertFalse("One string", result.size() <= 1);

} catch (Exception ex) {

fail("Valid Id Failed " + ex);

}

}

@Test

public void testDeleteProductNormal() {

System.out.println("DeleteProduct");

Product item = new Product();

Random RND = new Random();

int code = RND.nextInt(1000) + 2000;

item.setCode(code);

item.setCount(3.4f);

item.setIsPacked(false);

item.setName("Abacaba");

item.setPrice(RND.nextInt(1000));

boolean expResult = true;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.AddProduct(item);

boolean result = SQLAgent.DeleteProduct(code);

assertEquals(expResult, result);

} catch (Exception ex) {

fail("Normal delete failed");

}

}

@Test

public void testDeleteProductNegativeCode() {

System.out.println("DeleteProduct");

boolean expResult = false;

try {

boolean result = SQLAgent.DeleteProduct(-61);

assertEquals(expResult, result);

} catch (Exception ex) {

fail("Negative code delete failed");

}

}

@Test

public void testDeleteProductNullCode() {

System.out.println("DeleteProduct");

boolean expResult = false;

try {

boolean result = SQLAgent.DeleteProduct(0);

assertEquals(expResult, result);

} catch (Exception ex) {

fail("Null code delete failed");

}

}

@Test

public void testDeleteProductInvalid() {

System.out.println("DeleteProduct");

boolean expResult = false;

try {

boolean result = SQLAgent.DeleteProduct(1234568);

assertEquals(expResult, result);

} catch (Exception ex) {

fail("Invalid delete failed");

}

}

@Test

public void testAddProductNormal() {

System.out.println("AddProduct");

Product item = new Product();

Random RND = new Random();

item.setCode(RND.nextInt(1000) + 1000);

item.setCount(3.4f);

item.setIsPacked(false);

item.setName("Abacaba");

item.setPrice(RND.nextInt(1000));

boolean expResult = true;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

boolean result = SQLAgent.AddProduct(item);

assertEquals(expResult, result);

} catch (Exception ex) {

fail("Normal product faild.");

}

}

@Test

public void testAddProductNegativeCount() {

System.out.println("AddProduct");

Product item = new Product();

Random RND = new Random();

item.setCode(RND.nextInt(1000) + 1000);

item.setCount(-3.4f);

item.setIsPacked(false);

item.setName("Abacaba");

item.setPrice(RND.nextInt(1000));

boolean expResult = false;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

boolean result = SQLAgent.AddProduct(item);

assertEquals(expResult, result);

} catch (Exception ex) {

fail("Negative Count product faild.");

}

}

@Test

public void testAddProductNegativeCode() {

System.out.println("AddProduct");

Product item = new Product();

Random RND = new Random();

item.setCode(-666999);

item.setCount(3.4f);

item.setIsPacked(false);

item.setName("Abacaba");

item.setPrice(RND.nextInt(1000));

boolean expResult = false;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

boolean result = SQLAgent.AddProduct(item);

assertEquals(expResult, result);

} catch (Exception ex) {

fail("Negative Code product faild.");

}

}

@Test

public void testAddProductNullName() {

System.out.println("AddProduct");

Product item = new Product();

Random RND = new Random();

item.setCode(RND.nextInt(1000) + 1000);

item.setCount(3.4f);

item.setIsPacked(false);

item.setName(null);

item.setPrice(RND.nextInt(1000));

boolean expResult = false;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

boolean result = SQLAgent.AddProduct(item);

assertEquals(expResult, result);

} catch (Exception ex) {

fail("Null name product faild.");

}

}

@Test

public void testAddProductNegativePrice() {

System.out.println("AddProduct");

Product item = new Product();

Random RND = new Random();

item.setCode(RND.nextInt(1000) + 1000);

item.setCount(3.4f);

item.setIsPacked(false);

item.setName(null);

item.setPrice(-RND.nextInt(1000));

boolean expResult = false;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

boolean result = SQLAgent.AddProduct(item);

assertEquals(expResult, result);

} catch (Exception ex) {

fail("Null name product faild.");

}

}

@Test(expected = NullPointerException.class)

public void testAddProductNull() throws Exception {

System.out.println("AddProduct");

Product item = null;

boolean expResult = false;

SQLAgent.LoadSettings();

SQLAgent.Connect();

boolean result = SQLAgent.AddProduct(item);

assertEquals(expResult, result);

}

@Test

public void testAddProductDublicate() {

System.out.println("AddProduct");

Product item = new Product();

Random RND = new Random();

item.setCode(999);

item.setCount(3.4f);

item.setIsPacked(false);

item.setName("Abacaba");

item.setPrice(RND.nextInt(1000));

boolean expResult = false;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.AddProduct(item);

boolean result = SQLAgent.AddProduct(item);

assertEquals(expResult, result);

} catch (Exception ex) {

fail("Dublicate product faild.");

}

}

@Test

public void testUpdateProductNormal() {

System.out.println("UpdateProduct");

Product item = new Product();

Random RND = new Random();

int rnd = RND.nextInt(1000) + 2000;

item.setCode(rnd);

item.setCount(3.4f);

item.setIsPacked(false);

item.setName("Abacaba");

item.setPrice(RND.nextInt(1000));

boolean expResult = true;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.AddProduct(item);

item.setPrice(RND.nextInt(1000) + 3000);

boolean result = SQLAgent.UpdateProduct(item);

assertEquals(expResult, result);

} catch (Exception ex) {

fail("Normal - failed");

}

}

@Test

public void testUpdateProductNegPrice() {

System.out.println("UpdateProduct");

Product item = new Product();

Random RND = new Random();

int rnd = RND.nextInt(1000) + 2000;

item.setCode(rnd);

item.setCount(3.4f);

item.setIsPacked(false);

item.setName("Abacaba");

item.setPrice(RND.nextInt(1000));

boolean expResult = true;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.AddProduct(item);

item.setPrice(-RND.nextInt(1000) + 3000);

boolean result = SQLAgent.UpdateProduct(item);

assertEquals(expResult, result);

} catch (Exception ex) {

fail("NegPrice - failed");

}

}

@Test

public void testUpdateProductNegCode() {

System.out.println("UpdateProduct");

Product item = new Product();

Random RND = new Random();

int rnd = RND.nextInt(1000) + 2000;

item.setCode(rnd);

item.setCount(3.4f);

item.setIsPacked(false);

item.setName("Abacaba");

item.setPrice(RND.nextInt(1000));

boolean expResult = false;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.AddProduct(item);

item.setCode(-987654);

boolean result = SQLAgent.UpdateProduct(item);

assertEquals(expResult, result);

} catch (Exception ex) {

fail("NegCode - failed");

}

}

@Test

public void testUpdateProductNegCount() {

System.out.println("UpdateProduct");

Product item = new Product();

Random RND = new Random();

int rnd = RND.nextInt(1000) + 2000;

item.setCode(rnd);

item.setCount(3.4f);

item.setIsPacked(false);

item.setName("Abacaba");

item.setPrice(RND.nextInt(1000));

boolean expResult = true;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.AddProduct(item);

item.setCount(-3.4f);

boolean result = SQLAgent.UpdateProduct(item);

assertEquals(expResult, result);

} catch (Exception ex) {

fail("Empty Name - failed");

}

}

@Test(expected = NullPointerException.class)

public void testUpdateProductNullName() throws Exception {

System.out.println("UpdateProduct");

Product item = new Product();

Random RND = new Random();

int rnd = RND.nextInt(1000) + 2000;

item.setCode(rnd);

item.setCount(3.4f);

item.setIsPacked(false);

item.setName("Abacaba");

item.setPrice(RND.nextInt(1000));

boolean expResult = true;

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.AddProduct(item);

item.setName(null);

boolean result = SQLAgent.UpdateProduct(item);

assertEquals(expResult, result);

}

@Test

public void testUpdateProductEmptyName() {

System.out.println("UpdateProduct");

Product item = new Product();

Random RND = new Random();

int rnd = RND.nextInt(1000) + 2000;

item.setCode(rnd);

item.setCount(3.4f);

item.setIsPacked(false);

item.setName("Abacaba");

item.setPrice(RND.nextInt(1000));

boolean expResult = true;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.AddProduct(item);

item.setName("");

boolean result = SQLAgent.UpdateProduct(item);

assertEquals(expResult, result);

} catch (Exception ex) {

fail("EmptyName - Failed");

}

}

@Test

public void testUpdateProductInvalidCode() {

System.out.println("UpdateProduct");

Product item = new Product();

Random RND = new Random();

int rnd = RND.nextInt(1000) + 2000;

item.setCode(rnd);

item.setCount(3.4f);

item.setIsPacked(false);

item.setName("Abacaba");

item.setPrice(RND.nextInt(1000));

boolean expResult = false;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.AddProduct(item);

item.setCode(11);

boolean result = SQLAgent.UpdateProduct(item);

assertEquals(expResult, result);

} catch (Exception ex) {

fail("Invalide Code - failed");

}

}

@Test(expected = NullPointerException.class)

public void testUpdateProductNullProd() throws Exception {

System.out.println("UpdateProduct");

Product item = new Product();

Random RND = new Random();

int rnd = RND.nextInt(1000) + 2000;

item.setCode(rnd);

item.setCount(3.4f);

item.setIsPacked(false);

item.setName("Abacaba");

item.setPrice(RND.nextInt(1000));

boolean expResult = false;

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.AddProduct(item);

item = null;

boolean result = SQLAgent.UpdateProduct(item);

assertEquals(expResult, result);

}

@Test

public void testFixConnected() {

System.out.println("Fix");

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Fix();

} catch (Exception ex) {

fail("Fix connected");

}

}

@Test

public void testFixDisconnected() {

System.out.println("Fix");

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

SQLAgent.Disconnect();

SQLAgent.Fix();

} catch (Exception ex) {

fail("Fix disconnected");

}

}

}

### SQLAgentTest\_1

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.junit.Test;

import static org.junit.Assert.\*;

public class SQLAgentTest\_1 {

public SQLAgentTest\_1() {

}

@Test

public void testLoadSettingsValid() {

System.out.println("LoadSettings");

FileWriter fw;

try {

fw = new FileWriter("DBProps.prop");

BufferedWriter bw = new BufferedWriter(fw);

bw.append("QQQ\n");

bw.append("QQQ\n");

bw.append("QQQ\n");

bw.append("QQQ\n");

bw.flush();

bw.close();

fw.flush();

fw.close();

boolean result = SQLAgent.LoadSettings();

assertTrue("Error - valid", result);

} catch (IOException ex) {

Logger.getLogger(SQLAgentTest.class.getName()).log(Level.SEVERE, null, ex);

}

}

@Test

public void testLoadSettingsInvalid() {

System.out.println("LoadSettings");

FileWriter fw;

try {

fw = new FileWriter("DBProps.prop");

BufferedWriter bw = new BufferedWriter(fw);

bw.append("QQQ\n");

bw.append("QQQ\n");

bw.flush();

bw.close();

fw.flush();

fw.close();

boolean result = SQLAgent.LoadSettings();

assertFalse("Error - invalid", result);

} catch (IOException ex) {

Logger.getLogger(SQLAgentTest.class.getName()).log(Level.SEVERE, null, ex);

}

}

@Test

public void testLoadSettingsNofile() {

System.out.println("LoadSettings");

File fw;

try {

fw = new File("DBProps.prop");

fw.delete();

boolean result = SQLAgent.LoadSettings();

assertFalse("Error - No File", result);

} catch (Exception ex) {

Logger.getLogger(SQLAgentTest.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

### TradingterminalSuite

import org.junit.After;

import org.junit.AfterClass;

import org.junit.Before;

import org.junit.BeforeClass;

import org.junit.runner.RunWith;

import org.junit.runners.Suite;

@RunWith(Suite.class)

@Suite.SuiteClasses({org.aba.tradingterminal.DistributionTest.class, org.aba.tradingterminal.GeneratorTest.class, org.aba.tradingterminal.SQLAgentTest.class, org.aba.tradingterminal.SQLAgentTest\_1.class})

public class TradingterminalSuite {

}

### WhiteDistributionTest

import org.junit.Test;

import static org.junit.Assert.\*;

public class WhiteDistributionTest {

public WhiteDistributionTest() {

}

@Test

public void testGetIntCount() {

System.out.println("GetIntCount");

float count = 5;

int result = 0;

try {

result = Distribution.GetIntCount(count);

assertTrue(result >= 0);

} catch (Exception ex) {

fail("GetIntCount - exception! count=" + count + " result=" + result);

}

}

@Test

public void testGetFloatCount() {

System.out.println("GetFloatCount");

float count = 5;

float min = 0.5f;

float result = 0;

try {

result = Distribution.GetFloatCount(count, min);

assertTrue(result >= 0);

assertTrue(result >= min);

} catch (Exception ex) {

fail("GetFloatCount - exception! count=" + count + " min=" + min + "result=" + result);

}

}

@Test

public void testGetBuyers() {

System.out.println("GetBuyers");

int time = 5;

int clients = 100;

double[] IdealSum = {1.9999999};

int[] RealSum = {0};

int result = 0;

try {

result = Distribution.GetBuyers(time, clients, IdealSum, RealSum);

assertTrue("IdealSum < RealSum", IdealSum[0] >= RealSum[0]);

assertEquals("RealSum!=0", 0, RealSum[0]);

assertTrue(result >= 0);

} catch (Exception ex) {

fail("GetBuyers - exception! time=" + time + " clients=" + clients + " IdealSum=" + IdealSum[0] + " RealSum=" + RealSum[0] + " result=" + result);

}

}

@Test

public void testGetBuyersMinus1() {

System.out.println("GetBuyers");

int time = -1;

int clients = -1;

double[] IdealSum = {-1};

int[] RealSum = {-1};

int result = 0;

try {

result = Distribution.GetBuyers(time, clients, IdealSum, RealSum);

assertTrue(result >= 0);

} catch (Exception ex) {

fail("GetBuyers - exception! time=" + time + " clients=" + clients + " IdealSum=" + IdealSum[0] + " RealSum=" + RealSum[0] + " result=" + result);

}

}

}

### WhiteGeneratorTest

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.junit.Test;

import static org.junit.Assert.\*;

import org.junit.Before;

public class WhiteGeneratorTest {

public WhiteGeneratorTest() {

}

@Before

public void setUp() {

try {

FileWriter fw;

fw = new FileWriter("DBProps.prop");

BufferedWriter bw = new BufferedWriter(fw);

bw.append("tradingterminal2\n");

bw.append("50.50.0.106:3306\n"); // Network address!!!

bw.append("kramer98489\n");

bw.append("4321\n");

bw.flush();

bw.close();

fw.flush();

fw.close();

} catch (IOException ex) {

Logger.getLogger(SQLAgentTest.class.getName()).log(Level.SEVERE, null, ex);

}

}

@Test

public void testCreateCart() {

System.out.println("CreateCart");

float[] cart = {0, 0, 0, 0, 0, 0, 0, 0};

int count = 4;

boolean negative = false;

try {

SQLAgent.LoadSettings();

SQLAgent.Connect();

Generator.CreateCart(cart, count);

for (int i = 0; i < cart.length; i++) {

if (cart[i] < 0) {

negative = true;

}

}

assertFalse("Some of products have negative value", negative);

} catch (Exception ex) {

fail("Unexpected exeption");

}

}

}

### WhiteWorkerTest

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.junit.Test;

import static org.junit.Assert.\*;

import org.junit.Before;

public class WhiteWorkerTest {

public WhiteWorkerTest() {

}

@Before

public void setUp() {

try {

FileWriter fw;

fw = new FileWriter("DBProps.prop");

BufferedWriter bw = new BufferedWriter(fw);

bw.append("tradingterminal2\n");

bw.append("50.50.0.106:3306\n"); // Network address!!!

bw.append("kramer98489\n");

bw.append("4321\n");

bw.flush();

bw.close();

fw.flush();

fw.close();

} catch (IOException ex) {

Logger.getLogger(SQLAgentTest.class.getName()).log(Level.SEVERE, null, ex);

}

}

@Test

public void testStartSim() {

System.out.println("StartSim");

int peoplecount = 100;

int goodscount = 4;

SQLAgent.LoadSettings();

SQLAgent.Connect();

Worker instance = new Worker();

try {

instance.StartSim(peoplecount, goodscount);

} catch (Exception ex) {

fail("Simulation exception");

}

}

}

### WhiteBox

import org.junit.runner.RunWith;

import org.junit.runners.Suite;

@RunWith(Suite.class)

@Suite.SuiteClasses({org.aba.tradingterminal.WhiteDistributionTest.class, org.aba.tradingterminal.WhiteGeneratorTest.class, org.aba.tradingterminal.WhiteWorkerTest.class})

public class WhiteBox {

}

Выявленные ошибки:  
testGetFloatCountBigVersa – данная ошибка была обнаружена тестированием в режиме чёрного ящика. Причина возникновения ошибки была выявлена при тестировании в режиме белого ящика. Данная ошибка не была устранена по причине того, что она происходит из-за не корректной работы библиотечной функции Math.round, которая при слишком больших значениях не выполняет округление, а приводит число к 2,147,483,647 с типом float, из-за чего ожидаемый результат не совпадает с действительным на несколько порядков.

testGetBuyers99999 - данная ошибка была обнаружена тестированием в режиме чёрного ящика. Причина возникновения ошибки не была установлена.

При работе программы в обычном режиме данные ошибки произойти не могут.

5)Оптимизация и рефакторинг кода.  
В ходе оптимизации были удалены 4 класса:Buyer, Stat, Terminal, Admin.  
Их функционал был передан оставшимся классам. Расчёт покупки ведётся в классе SQLAgent, в нём же производится проверка корректности. Список продуктов покупателя был изменён на массив чисел с плавающей запятой, в котором значение 0 обозначает, что товар под номером i не был приобретён. Список продуктов хранится в единственном экземпляре в классе SQLAgent и доступен всем классам. В место создания очереди покупателей была создана целочисленная переменная queue, в которой хранится длина очереди в данный момент времени. Когда переменная queue > 1, вызывается метод SQLAgent.Buyed , в котором производится перебор массива и рассчитывается общая стоимость покупки.

Все поля классов с модификатором доступа public были изменены на private и снабжены “геттерами” и “сеттерами”. Дублирующие части кода были вынесены в отдельные методы. После оптимизации производительность программы была увеличена в несколько раз по сравнению с первоначальной реализацией.

Результаты тестирования.  
В результате модульного тестирования в режиме «чёрного ящика» и «белого ящика», а так же в результате системного тестирования было выявлено 2 ошибки, не влияющие на работу программы в обычном режиме.  
Системное тестирование проводилось посторонними людьми. Было проведено около 200 тестов, во время которых программа ни разу не потребовала перезагрузки.

Метрики:  
IEEE Плотность дефектов = n/m  
где n=2 – число уникальных дефектов  
m=1240 – число строк кода

Плотность дефектов = 2/1240 = 1/620 = 0,1613 %

IEEE Надёжность работы = (S/N)^k = 1  
где N-число запусков программы  
S-число успешных запусков  
в k произвольных случаях будет корректный результат

IEEE Плотность отказов = x/m= 0  
где x – число уникальных отказов  
m – число строк кода

Вывод: в ходе данной лабораторной работы была произведена оптимизация, рефакторинг и тестирование разработанного ПО во второй лабораторной работе. Производительность программы была увеличена, были найдены незначительные ошибки, не приводящие к сбою программы.