МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ

"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

ЛАБОРАТОРНА РОБОТА №7

Обробка виключних ситуацій та тестування в

мові програмування Java

Виконав: студент 1 курсу

група: ІО-21

Кузьменко В. З.

Номер залікової книжки № 2114

Прийняв: Невдащенко М.В.

Дата:

Оцінка:

Київ - 2013

Тема: Обробка виключних ситуацій та тестування в мові програмування Java.

Мета: Здобуття навичок у використанні механізму обробки виключних ситуацій та написанні тестів для перевірки працездатності методів в мові програмування Java.

Завдання: Модифікувати класи з попередніх лабораторних робіт, щоб обробка виключних ситуацій виконувалась за допомогою стандартних засобів мови програмування Java. Створити власний клас обробки виключних ситуацій.

Опис програми: Клас обробник виключних ситуацій MyException викликається при неправильній ініціалізації початкових даних (від'ємна ціна, вага, відсутність імені). Два класи-тести тестують методи програми.

Код програми:

Головний клас

Клас MyCollection

***package*** *lab6;*

***import*** *java.util.ListIterator;*

***public******class*** *Lab6 {*

*/\*\**

*\* Головний метод*

*\**

*\** ***@param*** *args*

*\** ***@throws***

*\*/*

***public******static******void*** *main(String[] args) {*

*/\*\**

*\* створюємо різну амуніцію*

*\*/*

*Armor armor =* ***null****;*

*Shield shield =* ***null****;*

*Sword sword =* ***null****;*

*Helmet helmet =* ***null****;*

***try*** *{*

*armor =* ***new*** *Armor("armor1", 100, 20.5);*

*shield =* ***new*** *Shield("shield1", 90, 56.7);*

*sword =* ***new*** *Sword("sword1", 120, 10.8);*

*helmet =* ***new*** *Helmet("helmet1", 60, 3.8);*

*Helmet h =* ***new*** *Helmet(" msdf", -5, 10);*

*}* ***catch*** *(MyException e) {*

*e.printStackTrace();*

*}*

*/\*\**

*\* ствоорюємо масив амуніції для екіпіровки одного лицаря*

*\*/*

*MyCollection am =* ***new*** *MyCollection();*

***try*** *{*

*am.add(helmet);*

*am.add(shield);*

*am.add(armor);*

*am.add(sword);*

*}* ***catch*** *(NullPointerException e) {*

*System.out.println("null exception");*

*}* ***catch*** *(ArrayIndexOutOfBoundsException e2) {*

*System.out.println("ArrayIndexOutOfBoundsException");*

*}*

*ListIterator<Ammunition> itr = am.listIterator(0);*

*Helmet hel =* ***null****;*

***try*** *{*

*hel =* ***new*** *Helmet("sdfsdf", 20, 20);*

*}* ***catch*** *(MyException e) {*

*e.printStackTrace();*

*}*

***while*** *(itr.hasNext()) {*

*Ammunition ammunition = itr.next();*

***if*** *(ammunition !=* ***null****) {*

*System.out.println(ammunition.getInformation());*

*}*

*}*

*am.remove(helmet);*

*am.add(hel);*

*/\*\**

*\* перевіряємо чи набрана амуніція підходить для одного лицаря*

*\*/*

***if*** *(am.isCorrectAmmunition()) {*

*Chevalier chevalier =* ***new*** *Chevalier(am);*

*System.out.print("Вартість амуніції лицаря: ");*

*/\*\* рахуємо вартість амуніції лицаря \*/*

*System.out.println(chevalier.getPriceAmmunition());*

*/\*\* сортуємо за вагою \*/*

*chevalier.sortAmmunition(am);*

*System.out.println("Відсортована амуніція:");*

*ListIterator<Ammunition> itr1 = chevalier.sortAmmunition(am)*

*.listIterator();*

***while*** *(itr1.hasNext()) {*

*System.out.println(itr1.next().getInformation());*

*}*

*System.out.println();*

*}*

*/\*\* діапазон цін для амуніції \*/*

***int*** *minPrice = 15;*

***int*** *maxPrice = 100;*

*System.out.println("Aмуніція із вибраного діапазону цін: " + minPrice*

*+ "-" + maxPrice);*

*/\*\* пошук амуніції по заданому діапазоні \*/*

*itr = am.listIterator();*

***while*** *(itr.hasNext()) {*

*Ammunition ammunition = itr.next();*

***if*** *((ammunition.getPrice() <= maxPrice)*

*&& (ammunition.getPrice() >= minPrice)) {*

*System.out.println(ammunition.getInformation());*

*}*

*}*

*}*

*}*

Клас MyException

***package*** *lab6;*

*/\*\**

*\* моя виключна ситуація v1.0*

*\** ***@author*** *Lenovo*

*\**

*\*/*

***public******class*** *MyException* ***extends*** *Exception {*

***public*** *MyException() {*

*System.out.println("Невірні параметри амуніції за адресою");*

*}*

*}*

Клас Ammunition

***package*** *lab6;*

*/\*\**

*\* Батьківський клас для Амуніції*

*\** ***@author*** *Volodia Kuzmenko*

*\**

*\*/*

***public******class*** *Ammunition {*

*/\*\*ціна \*/*

***private******int*** *price;*

*/\*\*вага\*/*

***private******double*** *weight;*

*/\*\*назва\*/*

***private*** *String name\_am;*

*/\*\**

*\**

*\** ***@param*** *name назва амуніції*

*\** ***@param*** *p ціна амуніції*

*\** ***@param*** *w вага амуніції*

*\** ***@throws*** *MyException*

*\*/*

***public*** *Ammunition() {*

*}*

***public*** *Ammunition (String name,* ***int*** *p ,* ***double*** *w)* ***throws*** *MyException {*

***if*** *(p <0 | w<0 | name ==* ***null****){*

***throw******new*** *MyException();*

*}*

*name\_am=name;*

*price=p;*

*weight=w;}*

*;*

*/\*\**

*\**

*\** ***@return*** *ціну*

*\*/*

***public******int*** *getPrice(){*

***return*** *price;*

*}*

*/\*\**

*\**

*\** ***@return*** *вагу*

*\*/*

***public******double*** *getWeight(){*

***return*** *weight;*

*}*

*/\*\**

*\**

*\** ***@return*** *назву амуніції*

*\*/*

***public*** *String getName() {*

***return*** *name\_am;*

*}*

*/\*\**

*\**

*\** ***@return*** *інформацію про предмет амуніції*

*\*/*

***public*** *String getInformation(){*

***return*** *"Name:" +name\_am+" Price:"+String.valueOf(price)+" Weight:"+String.valueOf(weight);*

*}*

*}*

Клас MyTestMycollection

***package*** *lab6;*

***import******static*** *org.junit.Assert.\*;*

***import*** *java.util.ArrayList;*

***import*** *org.junit.Before;*

***import*** *org.junit.Test;*

*/\*\**

*\* Клас-Тест для класу MyCollection*

*\** ***@author*** *Lenovo*

*\**

*\*/*

***public******class*** *MyTestMyCollection {*

***private*** *Ammunition am;*

***private*** *Ammunition am1;*

***private*** *Ammunition am2;*

***private*** *Ammunition am3;*

***private*** *Ammunition am4;*

*/\*\**

*\* метод буде запускатись перед кожним тестом*

*\** ***@throws*** *MyException*

*\*/*

*@Before*

***public******void*** *setParam()* ***throws*** *MyException{*

*am =* ***new*** *Ammunition("name0", 100, 200);*

*am1 =* ***new*** *Ammunition("name", 10, 20);*

*am2 =* ***new*** *Ammunition("name2", 30, 40);*

*am3 =* ***new*** *Ammunition("name2", 50, 60);*

*am4 =* ***new*** *Ammunition("name4", 70, 80);*

*}*

*/\*\**

*\* тест методу Remove на виключну ситуацію*

*\* NullPointerException*

*\*/*

*@Test (expected = NullPointerException.****class****)*

***public******void*** *testRemoveNullPointerException(){*

*MyCollection myCollection =* ***new*** *MyCollection();*

*myCollection.remove(****null****);*

*}*

*/\*\**

*\* тест методу Сontains на виключну ситуацію*

*\* NullPointerException*

*\*/*

*@Test (expected = NullPointerException.****class****)*

***public******void*** *testContainsNullPointerException(){*

*MyCollection col =* ***new*** *MyCollection();*

*col.contains(****null****);*

*}*

*/\*\**

*\* тест методу Remove на виключну ситуацію*

*\* IndexOutOfBoundsException*

*\*/*

*@Test (expected = IndexOutOfBoundsException.****class****)*

***public******void*** *testRemoveIndexOutOfBoundsException(){*

*MyCollection myCollection =* ***new*** *MyCollection();*

*myCollection.remove(-2);*

*myCollection.remove(8);*

*}*

*/\*\**

*\* тест методу Get на виключну ситуацію*

*\* IndexOutOfBoundsException*

*\*/*

*@Test (expected = IndexOutOfBoundsException.****class****)*

***public******void*** *testGetIndexOutOfBoundsException(){*

*MyCollection myCollection =* ***new*** *MyCollection();*

*myCollection.get(-5);*

*myCollection.get(35);*

*}*

*/\*\**

*\* тест методу Add на виключну ситуацію*

*\* NullPointerException*

*\*/*

*@Test (expected = NullPointerException.****class****)*

***public******void*** *testAddAmmunitionNullException(){*

*MyCollection myCollection =* ***new*** *MyCollection();*

*myCollection.add(****null****);*

*}*

*/\*\**

*\* Тест методу Add*

*\*/*

*@Test*

***public******void*** *testAddAmmunition() {*

*MyCollection myCollection =* ***new*** *MyCollection();*

*assertTrue(myCollection.add(am));*

*}*

*/\*\**

*\* тест методу addInt на виключну ситуацію*

*\* IndexOutOfBoundsException*

*\*/*

*@Test (expected = IndexOutOfBoundsException.****class****)*

***public******void*** *testAddArrayIndexOutOfBoundsException(){*

*MyCollection myCollection =* ***new*** *MyCollection();*

*myCollection.add(-5, am1);*

*}*

*/\*\**

*\* Тест методу addAll*

*\*/*

*@Test*

***public******void*** *testAddAllCollectionOfQextendsAmmunition() {*

*MyCollection col =* ***new*** *MyCollection();*

*ArrayList<Ammunition> arr =* ***new*** *ArrayList<Ammunition>();*

*arr.add(am); arr.add(am2);*

*assertTrue(col.addAll(arr));*

*}*

*/\*\**

*\* Тест методу AddAllIntCollection*

*\*/*

*@Test*

***public******void*** *testAddAllIntCollectionOfQextendsAmmunition() {*

*MyCollection col =* ***new*** *MyCollection();*

*ArrayList<Ammunition> arr =* ***new*** *ArrayList<Ammunition>();*

*arr.add(am); arr.add(am2);*

*assertTrue(col.addAll(1, arr));*

*assertTrue(col.get(0)==****null****);*

*assertTrue(col.get(2)==am2);*

*}*

*/\*\**

*\* Тест методу clear*

*\*/*

*@Test*

***public******void*** *testClear() {*

*MyCollection col =* ***new*** *MyCollection(am);*

*col.clear();*

*assertTrue(col.get(0)==****null****);*

*}*

*/\*\**

*\* Тест методу Contains*

*\*/*

*@Test*

***public******void*** *testContains() {*

*MyCollection col =* ***new*** *MyCollection();*

*assertTrue(col.indexOf(am)==-1);*

*col.add(am);*

*assertTrue(col.get(0)==am);*

*}*

*/\*\**

*\* Тест методу ContainsAll*

*\*/*

*@Test*

***public******void*** *testContainsAll() {*

*MyCollection col =* ***new*** *MyCollection(am);*

*ArrayList<Ammunition> arr =* ***new*** *ArrayList<Ammunition>();*

*arr.add(am);*

*assertTrue(col.containsAll(arr));*

*}*

*/\*\**

*\* Тест методу indexOf*

*\*/*

*@Test*

***public******void*** *testIndexOf() {*

*MyCollection col =* ***new*** *MyCollection();*

*assertTrue(col.indexOf(am)==-1);*

*col.add(am);*

*assertTrue(col.indexOf(am)==0);*

*}*

*/\*\**

*\* Тест методу isEmpty*

*\*/*

*@Test*

***public******void*** *testIsEmpty() {*

*MyCollection myCollection =* ***new*** *MyCollection();*

*assertTrue(myCollection.isEmpty());*

*}*

*/\*\**

*\* Тест методу LastIndexOf*

*\*/*

*@Test*

***public******void*** *testLastIndexOf() {*

*MyCollection col =* ***new*** *MyCollection();*

*col.add(am);*

*col.add(am);*

*assertTrue(col.lastIndexOf(am)==1);*

*}*

*/\*\**

*\* Тест методу Remove*

*\*/*

*@Test*

***public******void*** *testRemoveObject() {*

*MyCollection col =* ***new*** *MyCollection(am);*

*col.add(am1);*

*col.remove(am);*

*assertTrue(col.get(0)==am1);*

*}*

*/\*\**

*\* Тест методу RemoveInt*

*\*/*

*@Test*

***public******void*** *testRemoveInt() {*

*MyCollection col =* ***new*** *MyCollection(am);*

*col.add(am1);*

*col.remove(0);*

*assertTrue(col.get(0)==am1);*

*}*

*/\*\**

*\* Тест методу removeAll*

*\*/*

*@Test*

***public******void*** *testRemoveAll() {*

*Ammunition [] arr =* ***new*** *Ammunition[3];*

*arr[0]=am1;*

*arr[1]=am2;*

*arr[2]=am3;*

*MyCollection col =* ***new*** *MyCollection(arr);*

*ArrayList<Ammunition> amm =* ***new*** *ArrayList<Ammunition>();*

*amm.add(am1);*

*amm.add(am2);*

*col.removeAll(amm);*

*assertTrue(col.indexOf(am1)==-1);*

*assertTrue(col.get(0)==am3);*

*}*

*/\*\**

*\* Тест методу RetainsAll*

*\*/*

*@Test*

***public******void*** *testRetainAll() {*

*Ammunition [] arr =* ***new*** *Ammunition[3];*

*arr[0]=am1;*

*arr[1]=am2;*

*arr[2]=am3;*

*MyCollection col =* ***new*** *MyCollection(arr);*

*ArrayList<Ammunition> amm =* ***new*** *ArrayList<Ammunition>();*

*amm.add(am1);*

*col.retainAll(amm);*

*assertTrue(col.indexOf(am2)==-1);*

*assertTrue(col.get(0)==am1);*

*}*

*/\*\**

*\* Тест методу Set*

*\*/*

*@Test*

***public******void*** *testSet() {*

*MyCollection col =* ***new*** *MyCollection();*

*col.add(am1);*

*col.add(am3);*

*col.set(1, am2);*

*assertTrue(col.get(1)==am2);*

*}*

*/\*\**

*\* Тест методу size*

*\*/*

*@Test*

***public******void*** *testSize() {*

*MyCollection col =* ***new*** *MyCollection();*

*col.add(am1);*

*col.add(am3);*

*col.add(am2);*

*assertTrue(col.size()==3);*

*}*

*/\*\**

*\* Тест методу subList*

*\*/*

*@Test*

***public******void*** *testSubList() {*

*MyCollection col =* ***new*** *MyCollection();*

*col.add(am1);*

*col.add(am2);*

*col.add(am3);*

*col.add(am4);*

*ArrayList<Ammunition> ar1 =* ***new*** *ArrayList<Ammunition>();*

*ar1.add(am2);*

*ar1.add(am3);*

*ArrayList<Ammunition> ar2 = col.subList(1, 3);*

*assertArrayEquals(ar1.toArray(), ar2.toArray());*

*}*

*/\*\**

*\* Тест методу toArray*

*\*/*

*@Test*

***public******void*** *testToArray() {*

*MyCollection col =* ***new*** *MyCollection();*

*col.add(am1);*

*col.add(am2);*

*col.add(am3);*

*col.add(am4);*

*Ammunition [] am =* ***new*** *Ammunition [4];*

*am[0]=am1;*

*am[1]=am2;*

*am[2]=am3;*

*am[3]=am4;*

*assertArrayEquals(am, col.toArray());*

*}*

*/\*\**

*\* Тест методу IsCorrectAmmunition*

*\*/*

*@Test*

***public******void*** *testIsCorrectAmmunition() {*

*Helmet am1=****null****;*

*Armor am2=****null****;*

*Sword am3=****null****;*

*Shield am4=****null****;*

***try*** *{*

*am1 =* ***new*** *Helmet("name", 10, 20);*

*am2 =* ***new*** *Armor("name2", 30, 40);*

*am3 =* ***new*** *Sword("name2", 50, 60);*

*am4 =* ***new*** *Shield("name4", 70, 80);*

*}* ***catch*** *(MyException e) {*

*System.out.println("при таких параметрах це не буде виводитись");*

*e.printStackTrace();*

*}*

*MyCollection col =* ***new*** *MyCollection();*

*col.add(am1);*

*col.add(am2);*

*col.add(am3);*

*col.add(am4);*

*assertTrue(col.isCorrectAmmunition());*

*Helmet am5=****null****;*

***try****{*

*am5 =* ***new*** *Helmet("name5", 90 , 10);*

*}****catch****(MyException e){*

*}*

*col.add(am5);*

*assertFalse(col.isCorrectAmmunition());*

*}*

*}*

Клас MyTestMyListIterator

***package*** *lab6;*

***import******static*** *org.junit.Assert.\*;*

***import*** *java.util.ListIterator;*

***import*** *org.junit.Before;*

***import*** *org.junit.Test;*

*/\*\**

*\* Клас-тест для внутрішнього класу MyListIterator*

*\** ***@author*** *Lenovo*

*\**

*\*/*

***public******class*** *MyTestMyListIterator {*

***private*** *Ammunition am;*

***private*** *Ammunition am1;*

***private*** *Ammunition am2;*

***private*** *Ammunition am3;*

*/\*\**

*\* ініціалізація параметрів. Виконується перед кожним тестом*

*\** ***@throws*** *MyException*

*\*/*

*@Before*

***public******void*** *setParam()* ***throws*** *MyException{*

*am =* ***new*** *Ammunition("name0", 100, 200);*

*am1 =* ***new*** *Ammunition("name", 10, 20);*

*am2 =* ***new*** *Ammunition("name2", 30, 40);*

*am3 =* ***new*** *Ammunition("name2", 50, 60);*

*}*

*/\*\**

*\* Тест для конструктора MyListIteratorInt*

*\*/*

*@Test*

***public******void*** *testMyListIteratorInt() {*

*MyCollection col =* ***new*** *MyCollection(am);*

*ListIterator<Ammunition> itr = col.listIterator(1);*

*assertTrue(itr.nextIndex()==2);*

*assertTrue(itr.previous()==am);*

*}*

*/\*\**

*\* Тест для конструктора MyListIterator*

*\*/*

*@Test*

***public******void*** *testMyListIterator() {*

*MyCollection col =* ***new*** *MyCollection();*

*ListIterator<Ammunition> itr = col.listIterator();*

*assertFalse(itr.hasNext());*

*assertFalse(itr.hasPrevious());*

*}*

*/\*\*Тест для метода \*/*

*@Test*

***public******void*** *testAdd() {*

*MyCollection col =* ***new*** *MyCollection();*

*ListIterator<Ammunition> itr = col.listIterator();*

*itr.add(am);*

*assertTrue(col.get(0)==am);*

*}*

*/\*\*Тест для метода hasNext \*/*

*@Test*

***public******void*** *testHasNext() {*

*MyCollection col =* ***new*** *MyCollection(am);*

*ListIterator<Ammunition> itr = col.listIterator();*

*assertTrue(itr.hasNext());*

*itr.next();*

*itr.add(am);*

*assertFalse(itr.hasNext());*

*}*

*/\*\*Тест для метода hasPrevious \*/*

*@Test*

***public******void*** *testHasPrevious() {*

*MyCollection col =* ***new*** *MyCollection();*

*ListIterator<Ammunition> itr2 = col.listIterator();*

*assertFalse(itr2.hasPrevious());*

*col.add(am1);*

*col.add(am2);*

*itr2.next();*

*assertTrue(itr2.hasPrevious());*

*}*

*/\*\*Тест для метода next \*/*

*@Test*

***public******void*** *testNext() {*

*MyCollection col =* ***new*** *MyCollection(am1);*

*ListIterator<Ammunition> itr = col.listIterator();*

*assertTrue(itr.next()==am1);*

*}*

*/\*\*Тест для метода nextIndex \*/*

*@Test*

***public******void*** *testNextIndex() {*

*MyCollection col =* ***new*** *MyCollection(am1);*

*ListIterator<Ammunition> itr = col.listIterator();*

*assertTrue(itr.nextIndex()==1);*

*}*

*/\*\*Тест для метода previous \*/*

*@Test*

***public******void*** *testPrevious() {*

*MyCollection col =* ***new*** *MyCollection();*

*ListIterator<Ammunition> itr2 = col.listIterator();*

*assertTrue(itr2.previous()==****null****);*

*col.add(am1);*

*col.add(am2);*

*itr2.next();*

*assertTrue(itr2.previous()==am2);*

*}*

*/\*\*Тест для метода previous \*/*

*@Test*

***public******void*** *testPreviousIndex() {*

*MyCollection col =* ***new*** *MyCollection();*

*ListIterator<Ammunition> itr2 = col.listIterator();*

*assertTrue(itr2.previousIndex()==0);*

*col.add(am1);*

*col.add(am2);*

*itr2.next();*

*assertTrue(itr2.previousIndex()==1);*

*}*

*/\*\*Тест для метода remove \*/*

*@Test*

***public******void*** *testRemove() {*

*MyCollection col =* ***new*** *MyCollection();*

*ListIterator<Ammunition> itr2 = col.listIterator();*

*col.add(am1);*

*col.add(am2);*

*assertTrue(col.get(0)==am1);*

*assertTrue(col.get(1)==am2);*

*itr2.remove();*

*assertTrue(col.get(0)==am2);*

*assertTrue(col.get(1)==****null****);*

*}*

*/\*\*Тест для метода set \*/*

*@Test*

***public******void*** *testSet() {*

*MyCollection col =* ***new*** *MyCollection();*

*ListIterator<Ammunition> itr = col.listIterator();*

*itr.add(am1);*

*itr.add(am3);*

*itr.set(am2);*

*assertTrue(col.get(1)==am2);*

*}*

*}*

Висновки: я модифікував каси з попередніх лабораторних робіт, написав два тестові класи: MyTestMyListIterator і MyTestMycollection. В цих класах тестуються методи відповідних класів а також виключні ситуації які можуть виникнути при їх виконанні.