**Лабораторна робота №8**

**Тема:** Обробка виключних ситуацій та основи 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.

**Виконав:** студент І курсу

групи ІО-34

Куриленко Олександр

**Завдання:**

1. Модифікувати класи з попередніх лабораторних робіт (лабораторні роботи №6

та №7) таким чином, щоб обробка виключних ситуацій відбувалась за

допомогою стандартних засобів мови програмування Java. Створити власний

клас обробник виключних ситуацій.

2. Написати JUnit-тести для перевірки працездатності УСІХметодів та

виключних ситуацій.

3. Всі початкові дані задаються у виконавчому методі. Код повинен відповідати

стандартам JCC та бути детально задокументований.

**Код програми:**

**TestCollection**

**package** OOP.Lab8;

**import** OOP.lab6.\*;

**import** OOP.lab7.Array.MyCollection;

**import** junit.framework.Assert;

**import** org.junit.\*;

**import** java.util.Iterator;

/\*\*

\*

\* **@version** 1.0 4 June 2014

\* **@author** Kurilenko Alexander

\*

\* Класс TestCollection тестирует все методы класса MyCollection

\*/

**public** **class** TestCollection {

**private** MyCollection testCollection;

// инициирует testCollection для использования в тестах.

// Используется перед каждым тестом

@Before

**public** **void** initCollection() {

testCollection = **new** MyCollection();

**for** (**int** i = 0; i < 30; i++)

testCollection.add(**new** Ball(40, i, i, "blue"));

}

@Test

**public** **void** testConstructor() {

MyCollection collection = **new** MyCollection(**new** Ball(30, 5, 2, "Blue"));

Assert.assertNotNull(collection.get(0));

}

@Test

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

Assert.assertEquals(30, testCollection.size());

}

@Test

**public** **void** testEmpty() {

Assert.assertFalse(testCollection.isEmpty());

}

@Test

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

Assert.assertFalse(testCollection.contains(**new** Object()));

Assert.assertTrue(testCollection.contains(**new** Ball(40, 4, 4, "blue")));

}

@Test

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

Object[] temp = testCollection.toArray();

Toy[] toys = **new** Toy[40];

testCollection.toArray(toys);

**for** (**int** i = 0; i < testCollection.size(); i++)

Assert.assertEquals(toys[i], temp[i]);

}

@Test

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

Assert.assertTrue(testCollection.add(**new** Doll(40, 2, **false**)));

Assert.assertEquals(31, testCollection.size());

Assert.assertTrue(testCollection.contains(**new** Doll(40, 2, **false**)));

}

@Test

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

Assert.assertFalse(testCollection.remove(**null**));

Assert.assertTrue(testCollection.remove(**new** Ball(40, 4, 4, "blue")));

Assert.assertEquals(29, testCollection.size());

}

@Test

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

Assert.assertTrue(testCollection.indexOf(**null**) == -1);

Assert.assertTrue(testCollection.indexOf(**new** Ball(40, 4, 4, "blue")) > -1);

Assert.assertTrue(testCollection.lastIndexOf(**null**) == -1);

Assert.assertTrue(testCollection

.lastIndexOf(**new** Ball(40, 4, 4, "blue")) > -1);

}

@Test

**public** **void** testIterator() {

Iterator<Toy> iterator = testCollection.iterator();

Assert.assertTrue(iterator.hasNext());

**while** (iterator.hasNext()) {

Assert.assertNotNull(iterator.next());

}

}

@Test

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

MyCollection subCollection = (MyCollection) testCollection

.subList(0, 4);

Assert.assertTrue(subCollection.size() > 0);

Assert.assertTrue(testCollection.containsAll(subCollection));

}

@Test

**public** **void** testAddAll() {

MyCollection subCollection = (MyCollection) testCollection

.subList(0, 4);

Assert.assertTrue(testCollection.addAll(subCollection));

}

@Test

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

MyCollection subCollection = (MyCollection) testCollection

.subList(0, 4);

Assert.assertTrue(testCollection.removeAll(subCollection));

}

@Test

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

MyCollection subCollection = (MyCollection) testCollection

.subList(0, 4);

Assert.assertTrue(testCollection.retainAll(subCollection));

Assert.assertEquals(subCollection.size(), testCollection.size());

}

@Test

**public** **void** testGet() {

Assert.assertNotNull(testCollection.get(2));

}

@Test

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

testCollection.clear();

Assert.assertEquals(0, testCollection.size());

}

@Test

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

Assert.assertNotNull(testCollection.set(2, testCollection.get(1)));

}

}

**Висновок**

Дана лабораторна робота призначена для ознайомлення з обробкою виключних ситуацій та основ тестування в мові програмування Java.

Я здобув навички у використанні механізму обробки виключних ситуацій та написанні тестів для перевірки працездатності методів в мові програмування Java.