Міністерство освіти і науки України

Національний технічний університет

«Харківський політехнічний інститут»

Кафедра «Обчислювальна техніка та програмування»

ЗВІТ

Про виконання лабораторної роботи 16

“Модульне тестування”

Виконавець:

студент гр. КН-920В

Олексієнко Микита

Харків 2022

Лабораторна робота №16

**Модульне тестування**

**Мета**

Розробка модульних тестів з використанням JUnit 5.

**1.Вимоги**

**1.1 Розробник:**

Олексієнко Микита Віталійович

студент групи КН-920В

1. варіант

**1.2 Загальне завдання**

1. Розробити консольну програму для реалізації завдання обробки даних згідно прикладної області.
2. Для розміщення та обробки даних використовувати контейнери (колекції) і алгоритми з Java Collections Framework.
3. Забезпечити обробку колекції об'єктів: додавання, видалення, пошук, сортування згідно розділу Прикладні задачі л.р. №10.
4. Передбачити можливість довготривалого зберігання даних: 1) за допомогою стандартної серіалізації; 2) не використовуючи протокол серіалізації.
5. Продемонструвати розроблену функціональність в діалоговому та автоматичному режимах за результатом обробки параметрів командного рядка.

**2. Опис програми**

**2.1 Засоби ООП:**

У лабораторній роботі 16 , потрібно розробити тести до певних методів певним лабораторних робіт.

**2.2 Ієрархія та структура класів:**

- test

- alekseenko09

- ContainerTest.java

- alekseenko11

- ContainerTest.java

- alekseenko12

- ContainerTest.java

- alekseenko14

- ThreadsTest.java

**2.3 Важливі фрагменти програми:**

**alekseenko09.ContainerTest.java :**

package ua.khpi.oop.test.alekseenko09;  
  
import ua.khpi.oop.alekseenko09.Auto;  
import ua.khpi.oop.alekseenko09.Container;  
import ua.khpi.oop.alekseenko09.Container.ListIterator;  
  
import java.util.Arrays;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class ContainerTest {  
  
 @org.junit.jupiter.api.Test  
 void add() {  
 Container container = new Container();  
  
 container.add(new Auto("a1"));  
 container.add(new Auto("b2"));  
 container.add(new Auto("c3"));  
  
 assertEquals( 3, container.toArray().length );  
 }  
  
 @org.junit.jupiter.api.Test  
 void remove() {  
 Container container = new Container();  
  
 Auto forDelete = new Auto("b2");  
  
 container.add(new Auto("a1"));  
 container.add(forDelete);  
 container.add(new Auto("c3"));  
  
 Object[] arrayBefore = container.toArray();  
  
 container.remove(forDelete);  
  
 Object[] arrayAfter = container.toArray();  
  
 assertFalse(Arrays.equals(arrayBefore, arrayAfter));  
 }  
  
 @org.junit.jupiter.api.Test  
 void removeByPos() {  
 Container container = new Container();  
  
 container.add(new Auto("a1"));  
 container.add(new Auto("b2"));  
 container.add(new Auto("c3"));  
 container.add(new Auto("d4"));  
 container.add(new Auto("g5"));  
  
 int expectedPos = 3;  
  
 container.removeByPos(expectedPos);  
  
 Container container2 = new Container();  
  
 container2.add(new Auto("a1"));  
 container2.add(new Auto("b2"));  
 container2.add(new Auto("c3"));  
 container2.add(new Auto("g5"));  
  
 assertArrayEquals(container.toArray(), container2.toArray() );  
 }  
  
 @org.junit.jupiter.api.Test  
 void clear() {  
 Container container = new Container();  
  
 container.add(new Auto("a1"));  
 container.add(new Auto("b2"));  
 container.add(new Auto("c3"));  
  
 container.clear();  
 assertEquals(0, container.toArray().length );  
 }  
  
 @org.junit.jupiter.api.Test  
 void toArray() {  
 Container container = new Container();  
 assertTrue( container.toArray() instanceof Object[] );  
 }  
  
 @org.junit.jupiter.api.Test  
 void ContainerToString() {  
 Container container = new Container();  
 container.add(new Auto("a1"));  
  
 assertTrue( container.toString() instanceof String );  
 }  
  
 @org.junit.jupiter.api.Test  
 void contains() {  
 Container container = new Container();  
  
 container.add(new Auto("a1"));  
 container.add(new Auto("b2"));  
 container.add(new Auto("c3"));  
 container.add(new Auto("d4"));  
 container.add(new Auto("g5"));  
  
 container.contains(new Auto("c3") );  
 }  
  
 @org.junit.jupiter.api.Test  
 void xml() {  
 Container containerToRead = new Container();  
  
 containerToRead.add(new Auto("a1"));  
 containerToRead.add(new Auto("b2"));  
 containerToRead.add(new Auto("c3"));  
 containerToRead.add(new Auto("d4"));  
 containerToRead.add(new Auto("g5"));  
  
 final String file\_path = System.getProperty("user.dir") + "\\src\\ua\\khpi\\oop\\test\\alekseenko09\\xml.xml";  
  
 containerToRead.xml\_save( file\_path );  
  
 Container containerToWrite = new Container();  
  
 containerToWrite.xml\_load( file\_path );  
  
 assertArrayEquals(containerToRead.toArray(), containerToWrite.toArray() );  
 }  
  
 @org.junit.jupiter.api.Test  
 void standart() {  
 Container containerToRead = new Container();  
  
 containerToRead.add(new Auto("a1"));  
 containerToRead.add(new Auto("b2"));  
 containerToRead.add(new Auto("c3"));  
 containerToRead.add(new Auto("d4"));  
 containerToRead.add(new Auto("g5"));  
  
 final String file\_path = System.getProperty("user.dir") + "\\src\\ua\\khpi\\oop\\test\\alekseenko09\\standart.xml";  
  
 containerToRead.standard\_save( file\_path );  
  
 Container containerToWrite = new Container();  
  
 containerToWrite.standard\_load( file\_path );  
  
 assertArrayEquals(containerToRead.toArray(), containerToWrite.toArray() );  
 }  
  
 @org.junit.jupiter.api.Test  
 void iterator() {  
  
 Container container = new Container();  
  
 container.add(new Auto("a1"));  
 container.add(new Auto("b2"));  
 container.add(new Auto("c3"));  
 container.add(new Auto("d4"));  
 container.add(new Auto("g5"));  
  
 ListIterator iterator = container.iterator();  
  
 int iterations\_count = 0;  
  
 while ( iterator.hasNext() ) {  
 assertNotNull(iterator.next());  
 iterations\_count++;  
 }  
  
 assertEquals(iterations\_count, container.toArray().length );  
  
 Container container1 = new Container();  
  
 container1.add(new Auto("a1"));  
 container1.add(new Auto("b2"));  
 container1.add(new Auto("c3"));  
 container1.add(new Auto("d4"));  
 container1.add(new Auto("g5"));  
  
 ListIterator iterator1 = container1.iterator();  
  
 int expectedPos = 3, i = 0;  
  
 while ( iterator1.hasNext() ) {  
 if ( expectedPos == i++ ) {  
 iterator1.remove();  
 }  
 iterator1.next();  
 }  
  
 Container container2 = new Container();  
  
 container2.add(new Auto("a1"));  
 container2.add(new Auto("b2"));  
 container2.add(new Auto("c3"));  
 container2.add(new Auto("g5"));  
  
 assertArrayEquals(container1.toArray(), container2.toArray() );  
 }  
}

**alekseenko11.ContainerTest.java :**

package ua.khpi.oop.test.alekseenko11;  
  
import org.junit.jupiter.api.Test;  
import ua.khpi.oop.alekseenko11.Container;  
  
import java.io.IOException;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class ContainerTest {  
  
 @Test  
 void addingWithChecking() {  
 Container container = new Container();  
  
 String[] fields = {"TESLA", "2021", "0", "0", "true", "50000"};  
  
 assertTrue(container.addingWithChecking(fields));  
 }  
  
 @Test  
 void loadFromFile() throws IOException {  
  
 Container container = new Container();  
  
 assertEquals(0, container.toArray().length );  
  
 final String file\_path = System.getProperty("user.dir") + "\\src\\ua\\khpi\\oop\\test\\alekseenko11\\save\_lab\_11.txt";  
  
 assertTrue(container.loadFromFile(file\_path));  
 }  
}

**alekseenko12.ContainerTest.java :**

package ua.khpi.oop.test.alekseenko12;  
  
import org.junit.jupiter.api.Test;  
import ua.khpi.oop.alekseenko12.Container;  
  
import java.io.IOException;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class ContainerTest {  
  
 @Test  
 void findCar() throws IOException {  
 Container container = new Container();  
  
 final String file\_path = System.getProperty("user.dir") + "\\src\\ua\\khpi\\oop\\test\\alekseenko12\\save\_lab\_12.txt";  
  
 assertTrue(container.loadFromFile(file\_path));  
 assertTrue(container.findCar());  
 }  
}

**alekseenko14.ThreadsTest.java :**

package ua.khpi.oop.test.alekseenko13;  
  
import org.junit.jupiter.api.Test;  
import ua.khpi.oop.alekseenko13.Menu;  
  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Iterator;  
import java.util.List;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class ThreadsTest {  
  
 @Test  
 void ThreadTests() throws IOException {  
 assertEquals(0, checkIdUniqueness(Menu.CarContainerInit( 1 )));  
 }  
  
 private int checkIdUniqueness( List<Long> ids ) {  
 Object[] array = ids.toArray();  
  
 for ( int i = 0; i < ids.size(); i++ ) {  
 for ( int j = i + 1; j < ids.size(); j++ ) {  
  
 String current\_id = String.valueOf(array[i]);  
 String next\_id = String.valueOf(array[j]);  
  
 if (current\_id.equals(next\_id)) return -1;  
 }  
 }  
  
 return 0;  
 }  
}

**3. Висновок:**

Розроблено модульні тести з використанням JUnit 5.