Міністерство освіти інауки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра АСОІУ

ЗВІТ

про виконання комп’ютерного практикуму № 8

з дисципліни

“Основи програмної інженерії”

Тема: Робота з потоками в мові програмування Java.

Виконав:

студент групи ІП-51

Субцельний Олександр Володимирович

«10» травня 2016 р.

Перевірив:

Радченко Костянтин Олександрович

«11» травня 2016 р.

**Мета**: Здобуття навичок у використанні потоків в мові програмування Java

**Завдання**

1. Створити клас, що складається з методів збереження до файлу та зчитування з файлу колекції (лабораторна робота №6), як єдиного об’єкту; як послідовності об’єктів узагальненого класу (лабораторна робота №5); як послідовності об’єктів у вигляді тексту

2. Написати JUnit-тести для перевірки працездатності усіх методів та виключних ситуацій.

3. Код повинен відповідати стандартам JCC та бути детально задокументований.

**Код програми**

**main**

import com.sun.org.apache.xerces.internal.impl.xs.SchemaNamespaceSupport;

import java.util.Scanner;

public class main {

public static void main(String[] arg) {

NewSet arr = new NewSet();

Rock[] r = new Rock[5];

for (int i = 0; i < 2; i++)

r[i] = new Rock();

r[0].rocksong("Liebe", 4.08, "rock");

r[1].rocksong("Lust", 3.24, "rock");

Jazz[] j = new Jazz[5];

for (int i = 0; i < 1; i++)

j[i] = new Jazz();

j[0].jazzsong("BlueSuedeShoes", 5.26, "jazz");

Narodniy[] nar = new Narodniy[5];

for (int i = 0; i < 2; i++)

nar[i] = new Narodniy();

nar[0].narodniysong("narodnaya", 3.42, "folk");

nar[1].narodniysong("RusskieBabki", 3.42, "folk");

arr.add(r[0]); arr.add(r[1]);

arr.add(j[0]); arr.add(nar[0]);

arr.add(nar[1]);

System.out.println("We have");

arr.toArray();

Scanner case1 = new Scanner(System.in);

while (true) {

System.out.println("what to do\n" +

"1: add\n" +

"2:show all\n" +

"3:remove\n" +

"4:clear\n" +

"5:size\n" +

"6:contain\n" + "7:isempty?\n");

int case2 = case1.nextInt();

switch (case2) {

case 1: {

System.out.println("add songs?[true/false]");

Scanner sc = new Scanner(System.in);

boolean y = sc.nextBoolean();

if (y) {

System.out.println("Enter n");

int n = sc.nextInt();

Scanner sc1 = new Scanner(System.in);

Scanner sc2 = new Scanner(System.in);

Scanner sc3 = new Scanner(System.in);

String temp;

for (int i = 0; i < n; i++) {

int rindex = 2;

int jindex = 1;

int narindex = 2;

System.out.println("Enter style of comp " + (i + 1));

temp = sc2.nextLine();

if (temp.equals("rock")) {

r[rindex] = new Rock();

System.out.println("Enter title of comp " + (i + 1));

r[rindex].title = sc3.nextLine();

System.out.println("Enter length of comp " + (i + 1));

r[rindex].length = sc1.nextDouble();

r[rindex].style = temp;

arr.add(r[rindex]);

rindex++; }

if (temp.equals("jazz")) {

j[jindex] = new Jazz();

System.out.println("Enter title of comp " + (i + 1));

j[jindex].title = sc3.nextLine();

System.out.println("Enter length of comp " + (i + 1));

j[jindex].length = sc1.nextDouble();

j[jindex].style = temp;

arr.add(r[jindex]);

jindex++; }

if (temp.equals("folk")) {

nar[narindex] = new Narodniy();

System.out.println("Enter title of comp " + (i + 1));

nar[narindex].title = sc3.nextLine();

System.out.println("Enter length of comp " + (i + 1));

nar[narindex].length = sc1.nextDouble();

nar[narindex].style = temp;

arr.add(nar[narindex]);

narindex++; }} }

break; }

case 2: {

arr.toArray();

break; }

case 3: {

Scanner styl = new Scanner(System.in);

int styl1 = styl.nextInt();

try {

arr.remove(styl1);

} catch (Except ex) { }

break; }

case 4: {

arr.clear();

break; }

case 5: {

arr.size();

//arr.SortByStyle("rock");

break; }

case 6: {

boolean f = arr.contains(r[1]);

System.out.println(f); }

case 7: {

boolean f = arr.isEmpty();

System.out.println(f); }}}

case 8:{ try (BufferedReader reader = new BufferedReader(new FileReader(new File("/home/subtselnyi/IdeaProjects/lab5/src/input.txt")))) {

String line = reader.readLine();

String temp[] = line.split(" ");

if (temp[0].equals("rock")) {

r[rindex] = new Rock();

r[rindex].style = temp[0];

r[rindex].length = Double.parseDouble(temp[2]);

r[rindex].title = temp[1];

arr.add(r[rindex]);

rindex++; }

if (temp.equals("jazz")) {

j[jindex] = new Jazz();

j[jindex].style = temp[0];

j[jindex].length = Double.parseDouble(temp[2]);

j[jindex].title = temp[1];

arr.add(j[jindex]);

jindex++; }

if (temp.equals("folk")) {

nar[narindex] = new Narodniy();

nar[narindex].title = temp[1];

nar[narindex].length =Double.parseDouble(temp[2]);

nar[narindex].style = temp[0]; arr.add(nar[narindex]);

narindex++; }

} catch (IOException e) {

e.printStackTrace(); } }

case 9:{

arr.write(); } }

}}

COMPOSITION

import java.util.Scanner;

public class Composition {

public String title;

public double length;

public String style;

public void settitle(String title){

this.title = title; }

public void setlength(double length){

this.length = length; }

JAZZ(Narodniy-rock аналогічний )

public class Jazz extends Composition{

public void setstyle(String style){

this.style = style; }

public void jazzsong(String title,double length,String style){

settitle(title);

setlength(length);

setstyle(style); }}

NEWSET

import java.util.Collection;

import java.util.Iterator;

import java.util.Set;

import java.io.IOException;

import java.lang.Throwable;

public class NewSet implements Set<Composition>{

static int countCompositions=0;

Composition[] album = new Composition[10];

@Override

public int size() {

System.out.println(countCompositions);

return countCompositions; }

@Override

public boolean isEmpty() {

return (countCompositions==0); }

@Override

public boolean contains(Object o) {

boolean f=true;

for( int i = 0; i<countCompositions; i++) {

if (album[i].equals(o)) return true; f=false; } return false; }

@Override

public Object[] toArray() {

for (int i = 0; i<countCompositions; i++) {

System.out.println(album[i].title);

System.out.println(album[i].length);

System.out.println(album[i].style); }

return album; }

@Override

public <T> T[] toArray(T[] a) {

return (T[]) album; }

@Override

public boolean add(Composition composition) {

if (countCompositions<album.length) {

album[countCompositions] = composition;

countCompositions++;

} else {

Composition[] temp = album;

album = new Composition[(int) (album.length\*1.5)];

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

album[i]=temp[i]; }

album[countCompositions] = composition;

countCompositions++; }

return false; }

@Override

public boolean remove(Object o) {

return false; }

public boolean remove(int index) throws Except {

boolean f=true;

if (index>countCompositions){

f=false;

throw new Except("no index"); }

for (int i = index - 1, j = index; i < countCompositions; i++, j++) {

album[i] = album[j];

album[countCompositions - 1] = null;

countCompositions--; }

return true; }

@Override

public boolean containsAll(Collection<?> c) {

return false; }

@Override

public void clear() {

for (int i = 0; i<countCompositions; i++)

album[i] = null;

countCompositions=0; }

public void setCompositions(Composition o){

add(o); }

public void setCompositions(Collection<Composition> o){

for (Composition c : o){

add(c); } }

public void SortByStyle(String style){

for (int i = 0; i<countCompositions;i++){

if (album[i].style.equals(style)){

System.out.println(album[i].title); } } }

EXCEPT

public class Except extends Exception {

public Except() {

super(); }

public Except(String message) {

super(message);

System.out.println(message); }

public Except(String message, Throwable cause) {

super(message, cause); }

public Except(Throwable cause) {

super(cause); }

protected Except(String message, Throwable cause, boolean enableSuppression, boolean writableStackTrace) {

super(message, cause, enableSuppression, writableStackTrace); }}

NEWSETTEEST

import org.junit.Test;

import static org.junit.Assert.\*;

public class NewSetTest {

NewSet obj = new NewSet();

@Test

public void isEmpty() throws Exception {

boolean actual = obj.isEmpty();

assertTrue(actual); }

@Test

public void remove() throws Exception {

Rock r = new Rock();

r = new Rock();

r.rocksong("Liebe", 4.08, "rock");

obj.add(r);

boolean actual = obj.remove(1);

assertTrue(actual); }}

TestWriteRead

import org.junit.Test;

import java.io.\*;

public class TestWriteRead {

@Test

public void readerwriter() {

String line;

boolean actual;

try (BufferedWriter writer = new BufferedWriter(new FileWriter(new File("/home/subtselnyi/IdeaProjects/lab5/src/outputtest.txt")))) {

writer.write("text area");

} catch (IOException e) {

e.printStackTrace();

}

try (BufferedReader reader = new BufferedReader(new FileReader(new File("/home/subtselnyi/IdeaProjects/lab5/src/outputtest.txt")))) {

line = reader.readLine();

if (line.equals("text area")) {

actual = true;

assert (actual); }

} catch (IOException e) {

e.printStackTrace(); } }}

**Висновок**

Здобув навички у використанні потоків в мові програмування Java.В ході лабораторної роботи був реалізован СЕТ на їєрархії музичних композицій та написані власні класи для обробки виключних ситуації та клас з тестами для перевірки працездатності методів. Обєкти можна зчитувати з файлів та результат роботи записувати у вихідний файл.