Науменко Д.

КІТ 101.8а

Лабораторна робота №6. Серіалізація/десеріалізація об’єктів. Бібліотека класів користувача

Мета

* Тривале зберігання та відновлення стану об’єктів.
* Ознайомлення з принципами серіалізації/десеріалізації об’єктів.
* Використання бібліотек класів користувача.

Вимоги

-Реалізувати і продемонструвати тривале зберігання/відновлення раніше розробленого контейнера за допомогою серіалізації/десеріалізації.

-Продемонструвати послідовну та вибіркову обробку елементів розробленого контейнера за допомогою власного і отриманого за обміном службового класу.

-Реалізувати та продемонструвати порівняння, сортування та пошук елементів у контейнері.

Код програми :

Main.java

package lab6;

import lab3.lab3;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Iterator;

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws IOException, ClassNotFoundException {

ArrayList<String> arr = new ArrayList<String>(Arrays.asList("Machine", "Car", "Helicopter", "Hello", "Hi"));

Func str = new Func(arr);

while(true) {

System.out.print("1) Add word\n"

+

"2) Serialize\n" +

"3) Deserialize\n" +

"4) Sort\n" +

"5) Find\n" +

"6) Exit\n");

Scanner sc = new Scanner(System.in);

int input = Integer.parseInt(sc.nextLine());

switch (input) {

case 1:

System.out.print("put the word: ");

str.add(sc.nextLine());

break;

case 2:

FileOutputStream fileOutputStream = new FileOutputStream("temp.txt");

ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);

objectOutputStream.writeObject(str);

objectOutputStream.close();

break;

case 3:

FileInputStream fileInputStream = new FileInputStream("temp.txt");

ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);

str = (Func) objectInputStream.readObject();

objectInputStream.close();

break;

case 4:

str.sort();

Iterator iterator = str.iterator();

while (iterator.hasNext()){

System.out.print(iterator.next() + " ");

}

System.out.println();

break;

case 5:

System.out.print("put word: ");

System.out.println(str.find(sc.nextLine()));

break;

case 6:

return;

default:

break;

}

}

}

}

Func.java

package lab6;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.Iterator;

public class Func implements Serializable{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

private ArrayList<String> str;

public Func(ArrayList<String> str) {

this.str = str;

}

public ArrayList<String> getData() {

return str;

}

public void setData(ArrayList<String> str) {

this.str = str;

}

public void add(String string) {

this.str.add(string);

System.out.println("added word:" + string);

System.out.println("new array: "+ str);

}

public void clear() {

str.clear();

System.out.println("method clear: " + str);

}

public boolean remove(String string) {

this.str.add(string);

System.out.println("array before removing: " + str);

return str.remove(string);

}

public String toString()

{

StringBuilder build = new StringBuilder();

for(String iter1: this.str)

{

build.append(iter1);

}

System.out.println("string without previous word: " + build);

return build.toString();

}

public Object[] toArray() {

String[] mas = str.toArray(new String[str.size()]);

System.out.println("mass: " + Arrays.toString(mas));

return str.toArray();

}

public int size() {

int s = str.size();

System.out.println("size: " + s);

return s;

}

public Iterator<String> iterator() {

return new newIterator();

}

private class newIterator<String> implements Iterator{

int index;

@Override

public boolean hasNext() {

if(index < str.size()) {

return true;

}

return false;

}

@Override

public Object next() {

return str.get(index++);

}

}

public boolean contains(String string) {

newIterator iterator = new newIterator();

while(iterator.hasNext()) {

if(iterator.next().toString().equals(string)) {

return true;

}

}

return false;

}

public boolean containsAll(Func container) {

return this.str.equals(container.str);

}

public void sort(){

Collections.sort(str, String.CASE\_INSENSITIVE\_ORDER);

}

public int find(String str1){

int index = 0;

Iterator iterator = iterator();

while (iterator.hasNext()){

index++;

if (iterator.next().equals(str1)){

return index;

}

}

return -1;

}

}

Iterator.java

**package** lab6;

**public** **interface** Iterator {

**public** **boolean** hasNext();

**public** String next();

}

Робота програми :

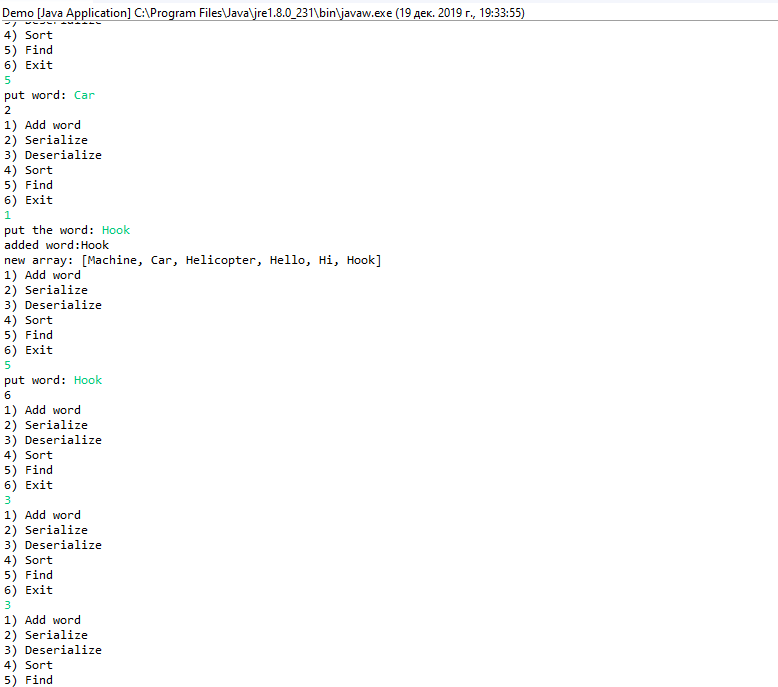


Рис. 1 – Робота програми

Ієрархія класів :

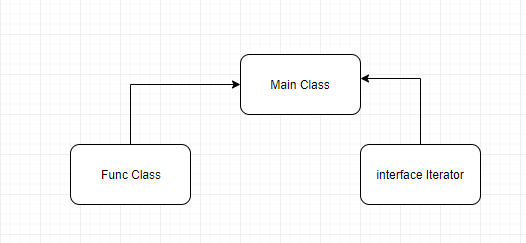


Рис.2 – Структура класів

Висновок : на лабораторній роботі ми ознайомились з принципами серіалізації/десеріалізації об’єктів.