Звіт

Автор: Пумня О., КІТ101.8б

Дата: 07.10.2019

Лабораторна робота №6

СЕРІАЛІЗАЦІЯ/ДЕСЕРІАЛІЗАЦІЯ ОБ'ЄКТІВ. БІБЛІОТЕКА КЛАСІВ КОРИСТУВАЧА

Мета. Ознайомлення з принципами серіалізації/десеріалізації об'єктів. Використання бібліотек класів користувача.

Вимоги:

1. Реалізувати і продемонструвати тривале зберігання та відновлення раніше розробленого контейнера за допомогою серіалізації та десеріалізації.
2. Обмінятися відкомпільованим (без початкового коду) службовим класом (Utility Class) рішення задачі л.р. №3 з іншим студентом (визначає викладач).
3. Продемонструвати послідовну та вибіркову обробку елементів розробленого контейнера за допомогою власного і отриманого за обміном службового класу.
4. Реалізувати та продемонструвати порівняння, сортування та пошук елементів у контейнері.
5. Розробити консольну програму та забезпечити діалоговий режим роботи з користувачем для демонстрації та тестування рішення.

ЗАВДАННЯ ДО РОБОТИ

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

ОПИС ПРОГРАМИ

Опис змінних

```
MyContainerMod container;     // модифікований контейнер
UserChioce choice;            // збереження вибору користувача
```

Ієрархія та структура класів

class Pumnya06 – точка входу в програму.

class MyContainerMod – модифікований клас-контейнер.

class UI – клас, що забезпечує діалоговий режим з користувачем.

ТЕКСТ ПРОГРАМИ

Текст файлу Pumnya05.java

```
package labs.pumnya06;

import labs.pumnya03.StrBuilder;
import labs.pumnya06.UI.UserChoice;
import labs.vasilchenko03.Helper;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.IOException;
import java.util.Comparator;

public final class Pumnya06 {
    private Pumnya06() {
    }
    /**
     * An entry point - main method.
     * @param args - arguments of main method
     * @throws IOException - input/output exceptions
     * @throws ClassNotFoundException - if class doesn't found
     */
    public static void main(final String[] args) throws IOException,
                                                        ClassNotFoundException {
        MyContainerMod container = new MyContainerMod();
        UserChoice choice;
        do {
            UI.mainMenu();
            choice = UI.enterChoice();
            System.out.println();
            if (choice == null) {
                System.out.println("An error has occurred.");
                continue;
            }
            switch (choice) {
                case AddValue:
                    System.out.println("Adding new value...");
                    System.out.print("Enter value: ");
                    container.add(UI.getString());
                    System.out.println();
                    break;
                case RemoveValue:
                    System.out.println("Removing value...");
                    System.out.print("Enter value: ");
                    if (container.remove(UI.getString())) {
                        System.out.println("Success.\n");
                    } else {
                        System.out.println("Value not found.\n");
                    }
                    break;
                case Clear:
                    System.out.println("Clearing...");
                    container.clear();
                    System.out.println("Done!\n");
                    break;
                case ShowAll:
                    System.out.println("All elements: ");
                    System.out.println(container.toString() + "\n");
                    break;
                case ContainCheck:
                    System.out.println("Checking for contain...");
                    System.out.print("Enter value: ");
                    if (container.contains(UI.getString())) {
                        System.out.println("Value contains in container.\n");
                    } else {
                        System.out.println("Value does not exist.\n");
                    }
                    break;
                case RunMyHelper:
                    System.out.print("Enter text: ");
                    String text = UI.getString();
                    System.out.print("Enter word: ");
                    String word = UI.getString();
```

```
                System.out.print("Enter sentence: ");
                String sentence = UI.getString();
                StrBuilder builder = new StrBuilder(text, word, sentence);
                container.add(text);
                container.add(word);
                container.add(sentence);
                container.add(builder.execute());
                System.out.println("Result: " + container.last() + "\n");
                break;
            case RunAnotherHelper:
                System.out.println("Enter string: ");
                String string = UI.getString();
                string = Helper.sortAlphabetical(string);
                System.out.println("Result: " + string + "\n");
                break;
            case Compare:
                System.out.println("Compare: ");
                container.compare();
                System.out.println();
                break;
            case SortAlphabetical:
                System.out.println("Sorting...");
                container.sort(String::compareTo);
                System.out.println("Result: "
                                   + container.toString() + "\n");
                break;
            case SortByLength:
                System.out.println("Sorting...");
                container.sort(Comparator.comparingInt(String::length));
                System.out.println("Result: "
                                   + container.toString() + "\n");
                break;
            case Search:
                System.out.print("Enter searched string: ");
                String searchedStr = UI.getString();
                System.out.println("Searching...");
                int[] strIndexes = container.search(searchedStr);
                for (int i : strIndexes) {
                    System.out.print(i + " ");
                }
                System.out.println();
                break;
            case SearchByLength:
                System.out.print("Enter searched length: ");
                int searchedLen = UI.getInt();
                System.out.println("Searching...");
                int[] intIndexes = container.search(searchedLen);
                for (int i : intIndexes) {
                    System.out.print(i + " ");
                }
                System.out.println();
                break;
            case Serialize:
                System.out.println("Serialization...");
                ObjectOutputStream oos = new ObjectOutputStream(
                                   new FileOutputStream("DataFile.dat"));
                oos.writeObject(container);
                oos.close();
                System.out.println("Done!\n");
                break;
            case Deserialize:
                System.out.println("Deserialization...");
                ObjectInputStream ois = new ObjectInputStream(
                                  new FileInputStream("DataFile.dat"));
                MyContainerMod getContainer =
                                (MyContainerMod) ois.readObject();
                ois.close();
                System.out.println(getContainer.toString() + "\n");
                break;
            case Exit:
                System.out.println("Exiting...");
                break;
            default:
                System.out.println("An error has occurred.");
        }
    } while (UI.getChoice() != UserChoice.Exit);
}
```

Текст файлу MyContainerMod.java

```
package labs.pumnya06;

import labs.pumnya05.MyContainer;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;

public class MyContainerMod extends MyContainer {
    /** Identifying key for serialization. */
    private static final long serialVersionUID = 6126733392129125019L;
    /**
     * Compare elements for equality.
     * Prints elements and their number.
     */
    public void compare() {
        ArrayList<String> arr = new ArrayList<>();
        ArrayList<Integer> arr2 = new ArrayList<>();
        int countOfEqual = 0;
        int temp = 0;
        for (String s : buffer) {
            if (!arr.contains(s)) {
                arr.add(s);
                arr2.add(++countOfEqual);
            } else {
                arr2.set(arr.indexOf(s),
                         arr2.get(arr.indexOf(s)) + 1);
            }
            countOfEqual = temp;
        }
        if (arr.size() == 0) {
            System.out.println("There are no equal elements");
        } else {
            for (int i = 0; i < arr.size(); i++) {
                System.out.println(arr.get(i) + ": " + arr2.get(i));
            }
        }
    }
    /**
     * Sorts elements of container.
     * @param comparator - type of sort
     */
    public void sort(final Comparator<String> comparator) {
        Arrays.sort(buffer, comparator);
    }
    /**
     * Search by given element.
     * @param string - searched string
     * @return an array of indexes of searched elements
     */
    public int[] search(final String string) {
        if (!this.contains(string)) {
            return null;
        }
        int size = 0;
        for (String s : buffer) {
            if (string.equals(s)) {
                size++;
            }
        }
        int[] searched = new int[size];
        int index = 0;
        for (int i = 0; i < buffer.length; i++) {
            if (string.equals(buffer[i])) {
                searched[index++] = i;
            }
        }
        return searched;
    }
    /**
     * Search by length of element.
     * @param length - length of element
     * @return an array of indexes of searched elements
     */
    public int[] search(final int length) {
        int size = 0;
```

```
        for (String s : buffer) {
            if (s.length() <= length) {
                size++;
            }
        }
        if (size == 0) {
            return null;
        }
        int[] searched = new int[size];
        int index = 0;
        for (int i = 0; i < buffer.length; i++) {
            if (buffer[i].length() <= length) {
                searched[index++] = i;
            }
        }
        return searched;
    }
}
```

Текст файлу UI.java

```
package labs.pumnya06;

import java.util.Scanner;

public final class UI {
    private UI() {
    }
    /** For getting data from keyboard. */
    private static Scanner in = new Scanner(System.in);
    /** For storing user choice. */
    private static int choice;
    /**  */
    private static UserChoice[] values = UserChoice.values();
    /**
     * Main menu text.
     */
    static void mainMenu() {
        System.out.println("01. Add value.");
        System.out.println("02. Remove value.");
        System.out.println("03. Clear.");
        System.out.println("04. Show all elements.");
        System.out.println("05. Check for contain.");
        System.out.println("06. Run my helper class.");
        System.out.println("07. Run friend's helper class.");
        System.out.println("08. Compare elements for equality.");
        System.out.println("09. Sort elements alphabetical.");
        System.out.println("10. Sort elements by length.");
        System.out.println("11. Search elements.");
        System.out.println("12. Search elements by length.");
        System.out.println("13. Serialize container.");
        System.out.println("14. Deserialize object from file.");
        System.out.println("00. Exit.");
        System.out.print("Enter here: ");
    }
    /**
     * Gets user choice and convert is to enum type.
     * @return converted from integer to enum type of user choice
     */
    public static UserChoice enterChoice() {
        getNumber();
        return (choice >= 0 && choice < values.length) ? values[choice] : null;
    }
    /**
     * Returns user choice.
     * @return user choice as enum type
     */
    public static UserChoice getChoice() {
        return (choice >= 0 && choice < values.length) ? values[choice] : null;
    }
    /**
     * Gets string from keyboard.
     * @return string gotten from keyboard
     */
    public static String getString() {
        return in.nextLine();
    }
```

```
    /**
     * Gets user choice from keyboard.
     */
    private static void getNumber() {
        choice = in.nextInt();
        in.nextLine();
    }
    /**
     * Gets integer from keyboard.
     * @return integer value
     */
    public static int getInt() {
        return in.nextInt();
    }
    public enum UserChoice {
        /** Exit from program. */
        Exit,
        /** Add value to container. */
        AddValue,
        /** Remove value from container. */
        RemoveValue,
        /** Clear container. */
        Clear,
        /** Show all elements of container. */
        ShowAll,
        /** Check element for contains to container. */
        ContainCheck,
        /** Run my helper class. */
        RunMyHelper,
        /** Run another helper class. */
        RunAnotherHelper,
        /** Comparing elements of container. */
        Compare,
        /** Alphabetical sorting. */
        SortAlphabetical,
        /** Sorting by length. */
        SortByLength,
        /** Searching for elements. */
        Search,
        /** Searching for elements by length. */
        SearchByLength,
        /** Serialization. */
        Serialize,
        /** Deserialization. */
        Deserialize
    }
}
```

ВАРІАНТИ ВИКОРИСТАННЯ

```
01. Add value.
02. Remove value.
03. Clear.
04. Show all elements.
05. Check for contain.
06. Run my helper class.
07. Run friend's helper class.
08. Compare elements for equality.
09. Sort elements alphabetical.
10. Sort elements by length.
11. Search elements.
12. Search elements by length.
13. Serialize container.
14. Deserialize object from file.
00. Exit.
```

Рисунок 1 – Діалогове меню

```
01. Add value.
02. Remove value.
03. Clear.
04. Show all elements.
05. Check for contain.
06. Run my helper class.
07. Run friend's helper class.
08. Compare elements for equality.
09. Sort elements alphabetical.
10. Sort elements by length.
11. Search elements.
12. Search elements by length.
13. Serialize container.
14. Deserialize object from file.
00. Exit.
Enter here: 4

All elements:
Hi, my name is Alex
```

Рисунок 2 – Виведення усіх елементів контейнера

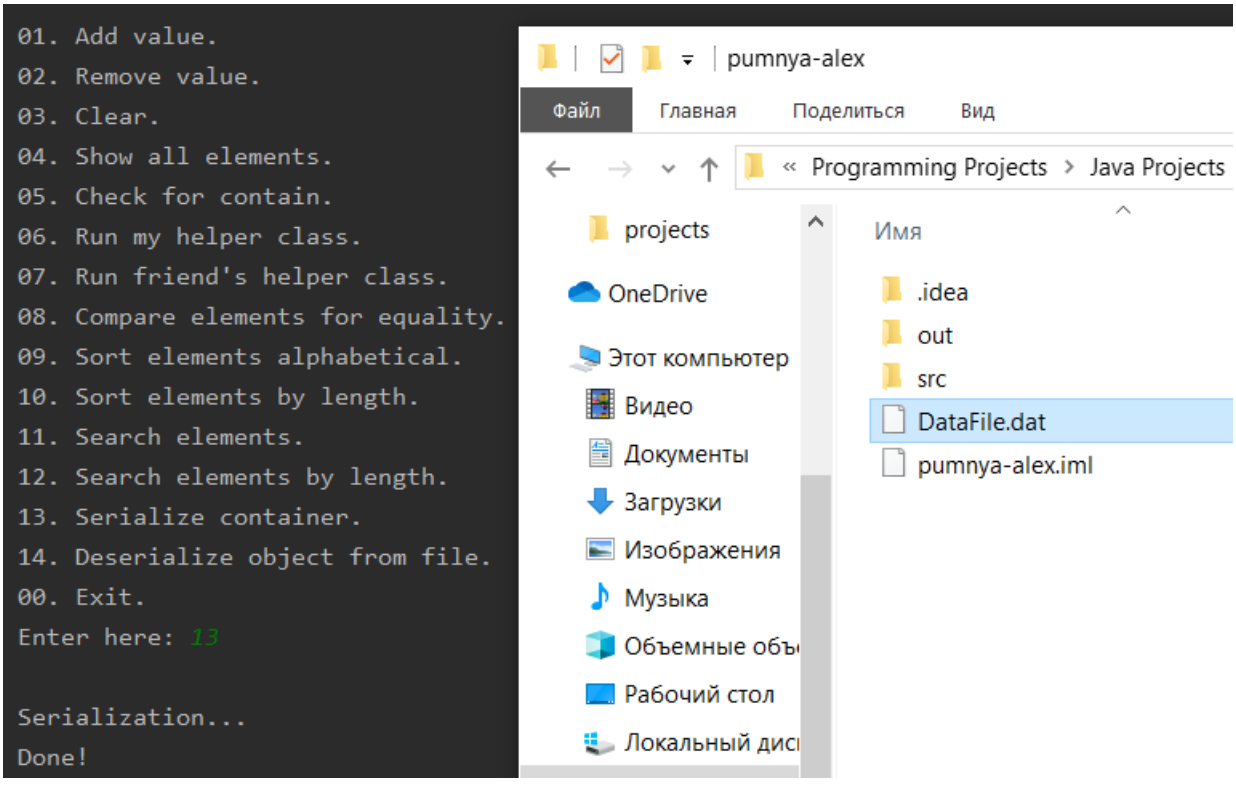

Рисунок 3 – Виконання серіалізації

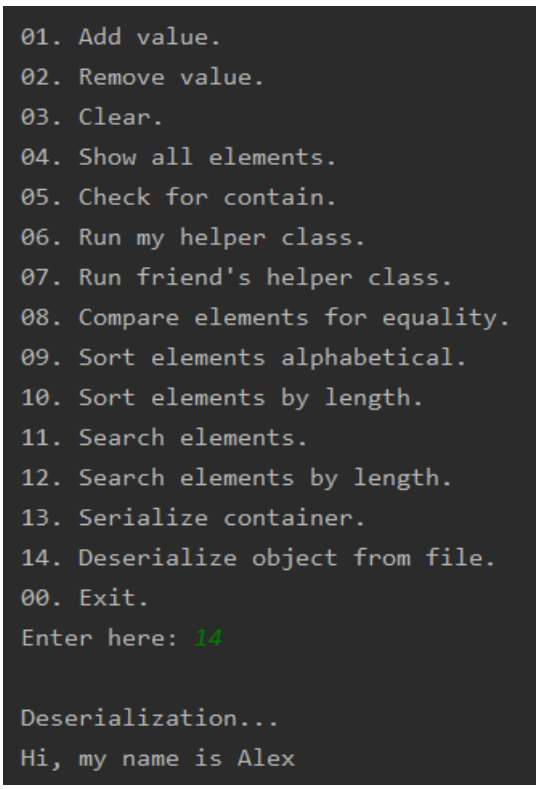

Рисунок 4 – Виконання десеріалізації

Програма може використовуватись як контейнер для об’єктів типу String. Реалізовано тривале збереження та відновлення контейнера. Також є можливість ітерування по контейнеру.

ВИСНОВКИ

При виконанні лабораторної роботи набуто практичних навичок щодо реалізації тривалого зберігання та відновлення даних за допомогою серіалізації та десеріалізації.