Тема: 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..

**1 ВИМОГИ**

**1.1 Розробник**

Інформація про розробника:

- Резніченко Дмитро Володимирович

- КІТ-319;

- 6 варіант.

**1.2 Загальне завдання**

1. Реалізувати і продемонструвати тривале зберігання/відновлення [раніше розробленого контейнера](https://oop-khpi.gitlab.io/%D0%B7%D0%B0%D0%B2%D0%B4%D0%B0%D0%BD%D0%BD%D1%8F/task05/) за допомогою [серіалізації/десеріалізації](https://docs.oracle.com/javase/8/docs/technotes/guides/serialization/index.html).
2. Обмінятися відкомпільованим (без початкового коду) службовим класом (Utility Class) рішення [задачі л.р. №3](https://oop-khpi.gitlab.io/%D0%B7%D0%B0%D0%B2%D0%B4%D0%B0%D0%BD%D0%BD%D1%8F/task03/#_4) 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.

1.3 Задача

Поновити попередню роботу

2 ОПИС ПРОГРАМИ

2.1 Засоби ООПРозробка класу Серіалізаціі/десеріалізаціі .  
2.2 Ієрархія та структура класівКлас “Helper ” виконує роль допоміжного класу який виконує неосновні завдання наприклад : виведення результату або перевірка символів на відповідність. Клас-контейнер «Сontainer» зберігає всі дані в масиві та надає доступ до даних . Методи класу : додавання , видалення , пошук, кількість елементів.

Ітератор «Iterator» - засобіб послідовного доступу до вмісту контейнера; він є інтелектуальним вказівником, що «знає» як отримати доступ до елементів контейнера; Serializator- клас розроблений для застосування тривале зберігання/ відновлення [раніше розробленого контейнера](https://oop-khpi.gitlab.io/%D0%B7%D0%B0%D0%B2%D0%B4%D0%B0%D0%BD%D0%BD%D1%8F/task05/) за допомогою [серіалізації/ десеріалізації](https://docs.oracle.com/javase/8/docs/technotes/guides/serialization/index.html) . Console\_program – розроблений клас для створення діалогового меню

2.3 Важливі фрагменти програми

/\*\*

\* **@author** Дмитро

/////////////////////////////////////// **import java.io.File;**

**import java.io.FileInputStream;**

**import java.io.FileNotFoundException;**

**import java.io.FileOutputStream;**

**import java.io.IOException;**

**import java.io.ObjectInputStream;**

**import java.io.ObjectOutputStream;**

**import java.lang.reflect.InaccessibleObjectException;**

**public class Serializator {**

**public boolean serialization(Сontainer container )**

**{**

**boolean flag=false;**

**File file = new File( "save.data");///pathname**

**ObjectOutputStream oos=null;**

**try {**

**FileOutputStream fos=new FileOutputStream(file);**

**if(fos!=null) {**

**oos= new ObjectOutputStream(fos);**

**oos.writeObject(container);**

**flag=true;**

**}**

**} catch(FileNotFoundException e) {e.printStackTrace();}**

**catch (IOException e) {e.printStackTrace(); }**

**finally {**

**if(oos!=null)**

**{try {oos.close();} catch (IOException e) {e.printStackTrace();}}**

**}**

**return flag;**

**}**

**public Сontainer deserializtion() {**

**Сontainer container=null;**

**File file = new File( "save.data");///pathname**

**ObjectInputStream ois=null;**

**try {**

**FileInputStream fis=new FileInputStream(file);**

**if(fis!=null) {**

**ois= new ObjectInputStream(fis);**

**container=(Сontainer)ois.readObject();**

**}**

**} catch(FileNotFoundException e) {e.printStackTrace();}**

**catch (IOException e) {e.printStackTrace(); }**

**catch (ClassNotFoundException e) {e.printStackTrace(); }**

**finally {**

**//if(ois!=null)**

**{try {ois.close();} catch (IOException e) {e.printStackTrace();}}**

**}**

**return container;**

**}**

**}**

**///////////////////////////////////////////////////////**

**package Laba6;**

**import java.io.Serializable;**

**import java.util.Scanner;**

**public class Сontainer implements Serializable {**

**private static final long serialVersionUID = 1L;**

**public static Scanner in = new Scanner(System.in);**

**private int size=0;**

**String[] m\_data=new String[255];**

**public String toString() //повертає вміст контейнера у вигляді рядка**

**{**

**if (size ==-1) {**

**System.out.print("\nМасив пустий. Елементів немає. Попернуто null");**

**return null;**

**}**

**String temp=new String();**

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

**temp+=m\_data[i];**

**return temp;**

**}**

**void add(String string) //додає вказаний елемент до кінця контейнеру;**

**{**

**if (size + 1 >= 255) return;**

**//m\_data[size++]=string;**

**String temp=new String();**

**for (int i=0;i<string.length();i++)**

**{**

**if((char)string.charAt(i)!=32)**

**temp+=string.charAt(i);**

**else{**

**m\_data[size++]=temp+" ";**

**temp=new String();**

**}**

**}**

**m\_data[size++]=temp+" ";**

**}**

**void clear()/// видаляє всі елементи з контейнеру;**

**{**

**while (size!=0)**

**iterator().remove();**

**}**

**boolean remove (String string)// видаляє перший випадок вказаного елемента з контейнера;**

**{**

**if (size ==0) return false;**

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

**if (m\_data[i]==string)**

**{ for (; i < size-1; i++)**

**m\_data[i]=m\_data[i+1];**

**this.m\_data[--size] = null;**

**return true;**

**}**

**return false;**

**}**

**int size()/// повертає кількість елементів у контейнері;**

**{**

**return this.size;**

**}**

**boolean contains(String string)/// повертає true, якщо контейнер містить вказаний елемент;**

**{**

**if (size ==0) return false;**

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

**if (m\_data[i]==string)**

**return true;**

**return false;**

**}**

**Object[] toArray() ///повертає масив, що містить всі елементи у контейнері;**

**{**

**return m\_data;**

**}**

**boolean containsAll(Сontainer container)/// повертає true, якщо контейнер містить всі елементи з зазначеного у параметрах;**

**{**

**if(container.size==size)**

**if(container.m\_data==m\_data)**

**return true;**

**return false;**

**}**

**void posuk(Сontainer container)//// помилку пошуку**

**{**

**if (container.size==0) {**

**System.out.print("Масив пустий");**

**return ;**

**}**

**System.out.print("Введіть шукане значення: ");**

**String text=new String();**

**text=in.nextLine();**

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

**if (container.m\_data[i].equals(text))**

**{**

**System.out.print("Позиція вашаго елемента: "+i);**

**return ;**

**}**

**System.out.print("Позиція вашаго елемента: незнайдена");**

**}**

**public void Sort(Сontainer k) {**

**boolean pr; /// для проверки отсортирован ли массив**

**do**

**{**

**pr = false;**

**for (int i = 0; i < k.size()-1; i++)**

**if (comparison (k.m\_data[i] , k.m\_data[i + 1])==1)**

**{**

**String temp=k.m\_data[i];**

**k.m\_data[i]=k.m\_data[i+1];**

**k.m\_data[i+1]=temp;**

**pr = true;**

**}**

**} while (pr);**

**}**

**public void d\_comparison(Сontainer conteiner)**

**{**

**int a,b;**

**while(true)**

**{**

**System.out.print("\nведіть індекс першого елементу: ");**

**a=in.nextInt();**

**if (a<0||a>size-1) System.out.print("\nЕлементу з таким індексом неіснує. Спробуйте ще раз");**

**else break;**

**}**

**while(true)**

**{**

**System.out.print("\nведіть індекс другого елементу: ");**

**b=in.nextInt();**

**if (b<0||b>size-1) System.out.print("\nЕлементу з таким індексом неіснує. Спробуйте ще раз");**

**else break;**

**}**

**int temp =Сontainer.comparison(conteiner.m\_data[a],conteiner.m\_data[b]);**

**if (temp==1)System.out.print("Результат порівнянн: a>b" );**

**else if (temp==-1)System.out.print("Результат порівнянн: a<b" );**

**else System.out.print("Результат порівнянн: a=b");**

**}**

**//@SuppressWarnings("unused")**

**public static int comparison(String a,String b)**

**{**

**int len=0;**

**if(a.length()<b.length())len=a.length();**

**else len=b.length();**

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

**{**

**if (a.charAt(i)>b.charAt(i)) return 1;**

**if (a.charAt(i)<b.charAt(i)) return -1;**

**}**

**if(a.length()<b.length())return -1;**

**else if (a.length()>b.length()) return 1;**

**return 0;**

**}**

**public Iterator<String> iterator() ///повертає ітератор відповідно до Interface Iterable.**

**{**

**return new m\_Iterator();**

**}**

**public class m\_Iterator implements Iterator<String>**

**{**

**int index = 0;**

**public boolean hasNext() {**

**if(index<size)**

**return true;**

**return false;**

**}**

**public String next()**

**{**

**return m\_data [index++];**

**}**

**public void remove()**

**{**

**for (int i=index; i < size-1; i++)**

**m\_data[i]=m\_data[i+1];**

**m\_data[--size] = null;**

**//throw new UnsupportedOperationException("remove");**

**}**

**public String begin()**

**{**

**return m\_data[0];**

**}**

**public String end()**

**{**

**return m\_data[size];**

**}**

**}**

**}**

**//////////////////////////////////////**

**package Laba6;**

**import java.util.Scanner;**

**import Laba3.Helper;**

**import ua.khpi.oop.kogutenko03.HelperClassWithString;**

**public class Console\_program {**

**public static Scanner in = new Scanner(System.in);**

**public static Serializator serializator=new Serializator();**

**public static int dialog()**

**{**

**System.out.println("\n\n Оберіть команду:"**

**+"\n\*1-вивести поточнi записанi даннi"**

**+"\n\*2 -Додати данні"**

**+"\n\*3 -виконати основне завдання"**

**+"\n\*4 - сортувати"**

**+"\n\*5 - пошук"**

**+"\n\*6 - порівняння"**

**+"\n\*7 - зберегти дані(save)"**

**+"\n\*8 - завантажити дані(load)"**

**+"\n\*9 (del)-очистити данi"**

**+"\n\*10 (exit)-вийти"**

**+"\n\*11 - Вивести данні ( ЛР 3)"**

**+"\n\*12 - Додати данні ( ЛР 3) "**

**+"\n\*13 - Сортування (ЛР 3)"**

**+"\n\n ваша команда: ");**

**return in.nextInt();**

**}**

**public static void Menu() ///функціє проводить координування по можливостям програм**

**{**

**Сontainer s=new Сontainer();**

**while(true)///нескінченний цикл який дозволяє працювати програмі**

**{**

**int k=dialog();**

**try {**

**switch(k)///пошук введеної команди**

**{**

**case 1: My\_Helper.PrintLine(s.toString());**

**break;**

**case 2:**

**System.out.print("Введіть ваш текст: ");**

**String text=new String();**

**text=in.nextLine();**

**s.add(in.nextLine());**

**break;**

**case 3:s=My\_Helper.Task6(s);**

**break;**

**case 4: s.Sort(s);**

**break;**

**case 5 : s.posuk(s);**

**break;**

**case 6 :s.d\_comparison(s);**

**break;**

**case 7 :**

**System.out.print("\n\nЗбереження даних:"+ serializator.serialization(s));**

**break;**

**case 8 : s=serializator.deserializtion();**

**break;**

**case 9 :s.clear();**

**break;**

**case 10 :return ;**

**case 11:Helper.PrintLineS(s.toString());**

**break;**

**case 12:**

**String text2=Helper.AddText();**

**s.add(text2);**

**break;**

**case 13:Helper.SortLexicographically(s.m\_data);**

**break;**

**}}catch(Exception e) {**

**System.out.print("\n\n\nЩось пішло не так. Але тепер все добре!!\n\n");}**

**}**

**}**

**}**

**3 ВАРІАНТИ ВИКОРИСТАННЯ**

Програма пропонує користувачеві обрати один із 10 пунктів. Після чого виконує відповідні дії притаманні пункту роботи. При виборі пункту зберегти дані - програма зберігає дані записані у контейнері в save.data файл розташований у папці проекту. А при відновлені, дані зчитуються з файлу save.data і записуються у контейнер. У разі якщо щось було у контейнері то воно перезаписується

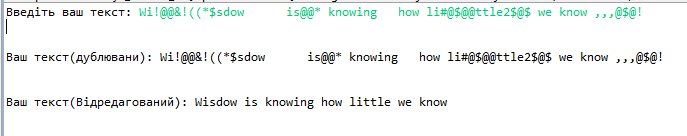


Рисунок 1 – результат редагування тексту

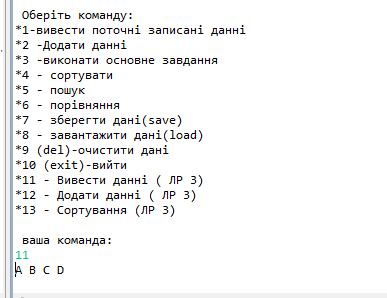


Рисунок 2 – меню

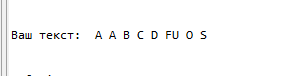


Рисунок 2 – Відсортований набір символів.

ВИСНОВКИ

Під час виконання лабораторної роботи було набуто навички роботи з принципами серіалізації/десеріалізації та роботі із відкомпільованими файлами .