**Серіалізація/десеріалізація об'єктів. Бібліотека класів користувача**

**Мета роботи**: Тривале зберігання та відновлення стану об'єктів.

Ознайомлення з принципами серіалізації/десеріалізації об'єктів.

Використання бібліотек класів користувача.

**Вимоги**

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/" \l "_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.

**Розробник**: Рябов Олексій Володимирович КІТ119а №18.

**Опис програми**

**Засоби ООП**: клас, метод классу.

**Структура класів:** один публічний клас Main, один утилітарний клас Task, один клас колекція Container

**Важливі фрагменти програми:**

import var18.Helper;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.util.Iterator;

import java.util.Scanner;

public class Main {

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

// Container container = new Container("Hi user.","This lab aims to show how I can deal with the container development problem.","All this is needed so that I can keep strings with palindromes safe and sound.");

Container container = new Container();

String command = "";

String str = "";

do

{

System.out.println("Доступні команди\r\n" +

"1 - заповнення контейнера данними - \r\n" +

"2 - виведення змісту контейнера - \r\n" +

"3 - додавання нового елементу в контейнер - \r\n" +

"4 - видалення елемнту з контейнера - \r\n" +

"5 - пошук елементу - \r\n" +

"6 - індивідуальне завдання - \r\n" +

"7 - сортування контейнеру - \r\n" +

"8 - порівняння контейнерів - \r\n" +

"9 - серіалізація - \r\n" +

"10 - десіалізація - \r\n" +

"11 - завдання іншого студента - \r\n" +

"0 - завершення програми(видалення контейнера автоматичне) - \r\n" +

"");

System.out.println("Введіть команду: ");

Scanner scan = new Scanner(System.in);

command = scan.nextLine();

switch (command) {

case "1": {

container = Create\_container();

break;

}

case "2": {

if(container.size()!=0)

{

System.out.println("Виведення змісту контейнера на екран");

System.out.println(container.toString());

}

else {

System.out.println("Ви ще не створили контейнер");

}

break;

}

case "3": {

if(container.size()!=0)

{

System.out.println("Введіть новий рядок");

str = scan.nextLine();

container.add(str);

}

else {

System.out.println("Ви ще не створили контейнер");

}

break;

}

case "4": {

if(container.size()!=0)

{

System.out.println("Введіть рядок що видалиться");

str = scan.nextLine();

System.out.println("Результат видалення елементу - " + container.remove(str));

}

break;

}

case "5": {

if(container.size()!=0)

{

System.out.println("Введіть рядок який потрібно перевірити");

str = scan.nextLine();

System.out.println("Результат пошуку елементу в контейнері" + container.contains(str));

}

else {

System.out.println("Ви ще не створили контейнер");

}

break;

}

case "6": {

if(container.size()!=0)

{

Task t = new Task();

t.start(container.toString());

}

else {

System.out.println("Ви ще не створили контейнер");

}

break;

}

case "7": {

if(container.size()!=0)

{

container.Sort();

}

else {

System.out.println("Ви ще не створили контейнер");

}

break;

}

case "8": {

if(container.size()!=0)

{

Container new\_cont = Create\_container();

container.containsAll(new\_cont);

}

else {

System.out.println("Ви ще не створили контейнер");

}

break;

}

case "9": {

FileOutputStream fs = new FileOutputStream("Serial.ser");

ObjectOutputStream os = new ObjectOutputStream(fs);

os.writeObject(container);

os.close();

System.out.println("Serialization successful");

break;

}

case "10": {

FileInputStream fis = new FileInputStream("Serial.ser");

ObjectInputStream ois = new ObjectInputStream(fis);

Container var = (Container) ois.readObject();

container = var;

ois.close();

break;

}

case "11": {

Helper.countVowelsAndConsonants(container.toString());

break;

}

case "0": {

System.out.println("Кінець програми");

container.clear();

break;

}

default:

System.out.println("Ви помилилися командой");

break;

}

}while(!(command.equals("0")));

}

public static Container Create\_container()

{

Scanner scan = new Scanner(System.in);

String enter\_data = "";

System.out.println("Введіть кількість елементів у контейнері");

enter\_data = scan.nextLine();

int size = Integer. parseInt(enter\_data);

System.out.println(""

+ "Введіть елементи - ");

String [] arr = new String[size];

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

System.out.print(i+1 + " елемент - " );

enter\_data = scan.nextLine();

arr[i] = enter\_data;

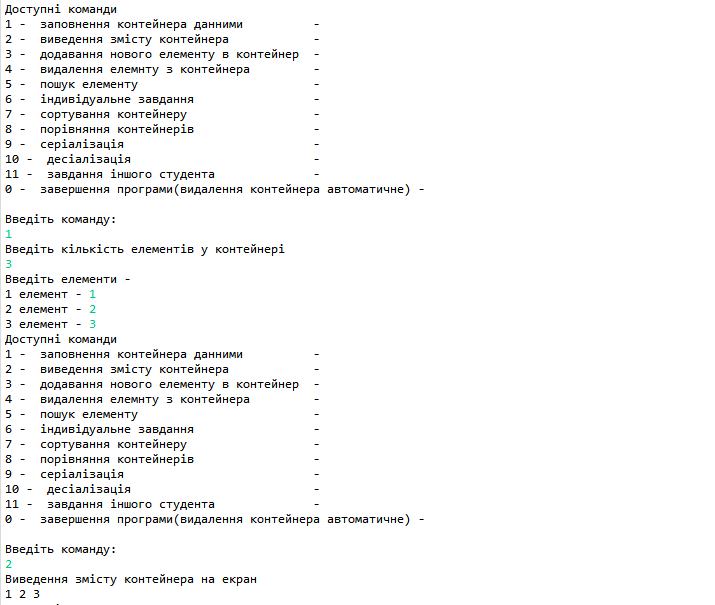
}

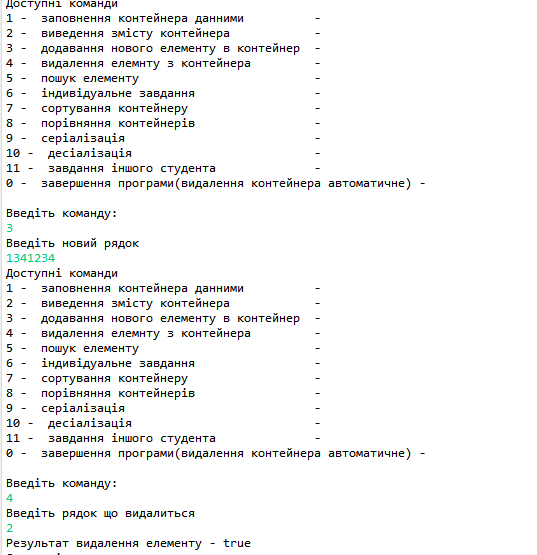
Container cont = new Container(arr);

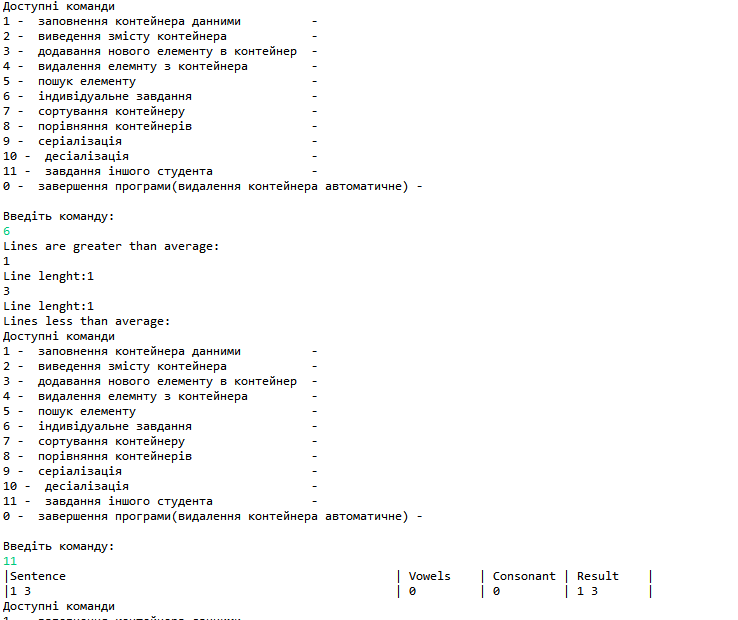
return cont;

}

**Результати роботи**



****

****

**Висновки**

Оволодів серіалізацією та десеріалізацією, навчився підключати сторонні бібліотеки та jar файли.