**Звіт**

**Лабораторна работа 6. 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 за допомогою серіалізації/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.
   1. **Розробник**: Момот Роман Євгенійович, КІТ119-а, варіант №14.
6. **ОПИС ПРОГРАМИ**
   1. **Засоби ООП**: клас, метод класу, поле класу.
   2. **Ієрархія та структура класів:** один публічний клас Main та публічний клас MyContainer, у полі якого знаходиться приватний класс MyIterator та публічний UtilityClass.
   3. **Важливі фрагменти програми:**

public class Main {

public static void main(String[] args) {

MyContainer array = new MyContainer();

Scanner scan = new Scanner(System.in);

int number;

boolean stop = false;

String choise;

String data;

while(stop != true)

{

System.out.println("Enter your choise");

System.out.println("1. Add data");

System.out.println("2. Output data");

System.out.println("3. Delete element");

System.out.println("4. Find element");

System.out.println("5. Personal task");

System.out.println("6. Sort data");

System.out.println("7. Compare arrays");

System.out.println("8. Serialize data");

System.out.println("9. Deserialize data");

System.out.println("10. Terminate program");

System.out.println("=====================");

System.out.print("Your choise: ");

choise = scan.nextLine();

switch(choise) {

case "1":

System.out.print("\nEnter your processed text: ");

array.add(scan.nextLine());

System.out.print("\n");

break;

case "2":

if(array.size() != 0)

{

System.out.println("\nData in array:");

for (int i = 0; i < array.size(); i++)

{

System.out.println(i+1 + ". " + array.getLine(i));

}

System.out.print("\n");

}

else

{

System.out.println("\nArray is empty.\n");

}

break;

case "3":

if(array.size() != 0)

{

System.out.print("\nEnter line to delete from array: ");

array.remove(scan.nextLine());

}

else

{

System.out.println("\nArray is empty.\n");

}

break;

case "4":

if(array.size() != 0)

{

System.out.print("Enter line to find in array: ");

if(array.contains(scan.nextLine()))

{

System.out.println("\nLine is found.\n");

}

else

{

System.out.println("\nLine not found.\n");

}

}

else

{

System.out.println("\nArray is empty.\n");

}

break;

case "5":

if(array.size() != 0)

{

for (int i = 0; i < array.size(); i++)

{

HelperClass.task(array.getLine(i));

System.out.print("\n");

}

}

else

{

System.out.println("\nArray is empty.\n");

}

break;

case "6":

if(array.size() != 0)

{

array.sort();

}

else

{

System.out.println("\nArray is empty.\n");

}

System.out.println("\nArray sorted\n");

break;

case "7":

if(array.size() != 0)

{

System.out.print("Enter number of lines in compared array: ");

number = scan.nextInt();

if(number >= 0)

{

MyContainer arrayToCompare = new MyContainer();

System.out.print("Enter your processed text: ");

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

{

System.out.print(i + ". ");

arrayToCompare.add(scan.nextLine());

System.out.println( );

}

}

}

else

{

System.out.println("\nArray is empty.\n");

}

break;

case "8":

if(array.size() != 0)

{

System.out.print("\nEnter file name: ");

data = scan.nextLine();

if(data.indexOf(".ser") == -1)

data += ".ser";

try

{

FileOutputStream file = new FileOutputStream(data);

ObjectOutputStream serial = new ObjectOutputStream(file);

serial.writeObject(array);

serial.close();

System.out.println("\nData serialized.\n");

}

catch(Exception ex)

{

System.out.println("\n" + ex.getMessage() + "\n");

}

}

else

{

System.out.println("\nArray is empty.\n");

}

break;

case "9":

System.out.print("\nEnter file name: ");

data = scan.nextLine();

if(data.indexOf(".ser") == -1)

data += ".ser";

try(ObjectInputStream ois = new ObjectInputStream(new FileInputStream(data)))

{

array = (MyContainer)ois.readObject();

ois.close();

}

catch(Exception ex){

System.out.println("\n" + ex.getMessage() + "\n");

}

break;

case "10":

stop = true;

break;

default:

System.out.println("Error. Wrong command.\n");

break;

}

}

System.out.println("\nTerminating the program.");

array.clear();

scan.close();

}

}

public class MyContainer implements Serializable {

private String[] array;

private int size;

public int size()

{

return size;

}

public String getLine(int index)

{

return array[index];

}

public String toString()

{

StringBuilder string = new StringBuilder();

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

string.append(array[i] + " ");

return string.toString();

}

public void add(String string)

{

String[] newArray = new String[size+1];

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

newArray[i] = array[i];

size++;

newArray[size - 1] = string;

array = newArray;

}

public void clear()

{

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

array[i] = null;

size = 0;

}

public boolean remove(String string)

{

boolean result = false;

int position = 0;

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

if(array[i].equals(string))

{

result = true;

position = i;

break;

}

if(result)

{

String[] newArray = new String[size-1];

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

newArray[i] = array[i];

for (int i = position; i+1 < size; i++)

newArray[i]=array[i+1];

size--;

array=newArray;

}

return result;

}

public Object[] toArray()

{

Object[] object = new Object[size];

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

object[i]=array[i];

return object;

}

public boolean contains(String string)

{

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

if (array[i].equals(string))

return true;

return false;

}

public boolean containsAll(MyContainer container)

{

boolean result = false;

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

{

result = false;

for (int j = 0; j < container.size(); j++)

{

if(array[i].equals(container.getLine(j)))

{

result = true;

break;

}

}

if(!result)

{

return false;

}

}

return result;

}

public MyContainer(String... strings)

{

if(strings.length > 0)

{

size = strings.length;

array = new String[size];

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

array[i]=strings[i];

}

}

public void sort() {

String temp;

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

{

for(int j = i + 1; j < array.length; j++)

{

if(array[i].compareTo(array[j]) > 0)

{

temp = array[i];

array[i] = array[j];

array[j] = temp;

}

}

}

}

public MyIterator<String> getIterator()

{

return new MyIterator<String>();

}

private class MyIterator<String> implements Iterator {

int index;

@Override public boolean hasNext()

{

if(index < size)

return true;

else

return false;

}

@Override public Object next()

{

if(index==size)

throw new NoSuchElementException();

return array[index++];

}

@Override public void remove()

{

MyContainer.this.remove(array[--index]);

}

}

}

**Висновки**

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи с серіалізацією та десеріалізацією об’єктів .

Програма протестована, виконується без помилок.