Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Інститут комп’ютерних наук та інформаційних технологій

Кафедра систем штучного інтелекту

Лабораторна робота №9

на тему:

“ Розробка власних контейнерів.
Ітератори. Серіалізація/десеріалізація об’єктів.
Бібліотека класів користувача.”

з курсу:

“Об’єктно-орієнтоване програмування ”

Виконала:
ст. гр. КН-110
Гелетій Софія
Прийняв:
Гасько Р.Т

Львів 2018

Мета:

- Набуття навичок розробки власних контейнерів.

● Використання ітераторів.

● Тривале зберігання та відновлення стану об’єктів.

● Ознайомлення з принципами серіалізації/десеріалізації об’єктів.

● Використання бібліотек класів користувача.

Вимоги

1. Розробити клас-контейнер, що і терується (docs.oracle.com/javase/8/docs/ api/java/lang/Iterable.html) для збереження початкових даних Вашого варіанту завдання з роботи №8 (Прикладні задачі. Список з 1-15 варіантів) у вигляді масиву рядків з можливістю додавання, видалення і зміни елементів.

2. В контейнері реалізувати та продемонструвати наступні методи:

String toString() повертає вміст контейнера у вигляді рядка;

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

void clear() видаляє всі елементи з контейнеру;

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

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

int size() повертає кількість елементів у контейнері;

boolean contains(String string) повертає t rue, якщо контейнер містить вказаний елемент;

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

public Iterator<String> iterator() повертає ітератор відповідно до Interface Iterab remove();

4. Продемонструвати роботу ітератора задопомогою циклів while иf or each.

5. Забороняється використання контейнерів (колекцій) і алгоритмів з J ava Collections Framework - https://docs.oracle.com/javase/8/docs/technotes/ guides/ collections/

6. Реалізувати і продемонструвати тривале зберігання/відновлення розробленого

контейнера за допомогою с еріалізації/десеріалізації. https://docs.oracle.com/javase/8/docs/technotes/guides/serialization/index.html

7. Обмінятися відкомпільованим (без початкового коду) службовим класом (Utility Class) рішення одного варіанту задачі (Прикладні задачі. Список з 1-15 варіантів) з сусіднім номером. 1 міняється з 2, 2 з 3, 3 з 4, 4 з 5 і т.д. Останній, 15 міняється з 1 варіантом і далі аналогічно.

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

9. Реалізувати та продемонструвати порівняння, сортування та пошук елементів

у контейнері.

10. Розробити консольну програму та забезпечити діалоговий режим роботи з користувачем для демонстрації та тестування рішення.

Використовується клас Iterator,Comparator для збереження контейнера масиву рядків та для маніпуляцій із даним масивом

Текст основних моментів програми

```
package com.company;
import java.io.Serializable;
import java.util.Arrays;
import java.util.Comparator;
public class Container implements Iterable <String>, Serializable
{
 /**
 * Class iteraror with methods hasnext,next
 *
```

```
*/
 class Iterator<String> implements java.util.Iterator <String>
{
 private int pointer;
 private final int end;
 public Iterator(java.lang.String[] data) {
 this.pointer = 0;
 this.end = data.length;
 }
 public boolean hasNext() {
 return this.pointer < end;
 }
 public String next() {
 if (!this.hasNext()) System.out.println("no such
elment");
 int current = pointer;
 pointer++;
 return (String) strings[current];
 }
 }
 //Iterator for strings
 public Iterator <String> iterator() {
 return new Iterator <String>(strings);
 }
 //Sort array by length
 final String[] sort() {
 Comparator <String> lenghtComparator = new Sort();
 String[] temp = strings.clone();
 Arrays.sort(temp, lenghtComparator);
 return temp;
```

```
}
public String[] strings;//array for strings
Object[] toArray() {//to array
return Arrays.copyOf(strings, strings.length);
}
int size() {//return size of array
return strings.length;
}
//Container construction
public Container(String[] array) {
```

```
strings = new String[array.length];
strings = array.clone();
}
public String toString() {
StringBuilder result = new StringBuilder();
for (String item : strings) {
result.append(item);
}
return result.toString();
}
//add element
void add(String string) {
String[] temp = new String[strings.length];
System.arraycopy(strings, 0, temp, 0, strings.length);
int newLength = strings.length + 1;
strings = new String[newLength];
System.arraycopy(temp, 0, strings, 0, temp.length);
strings[newLength - 1] = string;
}
```

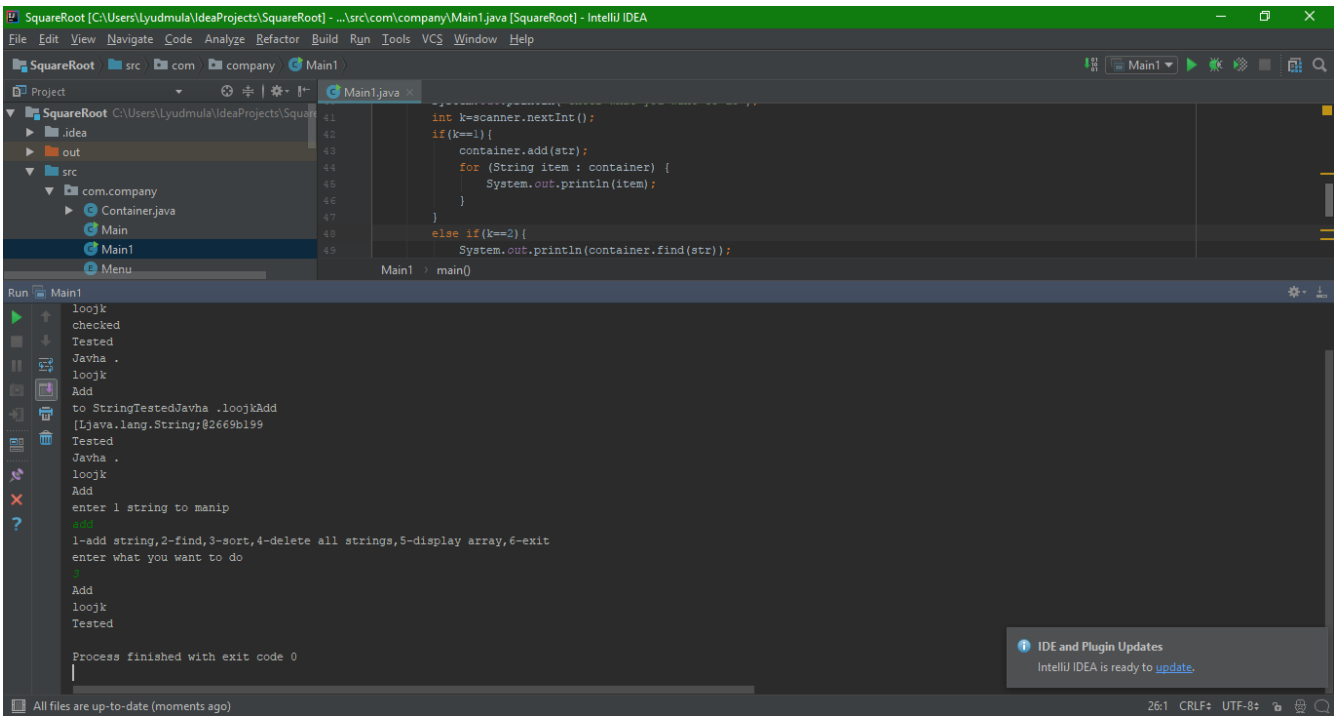
```
//clear container
void deleteAll() {
strings = new String[0];
}
//Delete one element
boolean removeOne(String string) {
for (int i = 0; i < strings.length; i++) {
if (strings[i].equals(string)) {
int removed = strings.length - i - 1;
if (removed > 0) {
System.arraycopy(strings, i + 1, strings, i,
removed);
}
strings[strings.length - 1] = null;
return true;
}
}
return false;
}
//find element
final boolean find(final String string) {
for (String item : strings) {
if (item.equals(string)) {
return true;
}
}
return false;
}
//check if array contains all elements in other array
final boolean containsAll(final Container container) {
```

```
 return Arrays.equals(this.strings, container.strings);
 }
}
//Class comparator for use in sort method
class Sort implements Comparator <String> {
public final int compare(final String o1, final String o2) {
 if (o1.length() > o2.length()) {
 return 1;
 } else {
 if (o1.length() < o2.length()) {
 return -1;
 } else {
 return 0;
 }
 }
 }
}
package com.company;
import java.util.Scanner;
public class Main {
 public static void main(String[] args) {
 Scanner scanner=new Scanner(System.in);
 String[] array = new String[3];
 array[0] = "Tested";
 array[1] = "Javha .";
 array[2] = "loojk";
 Container container = new Container(array);
 for (String item : container) {
 System.out.println(item);
 }
```

```
container.add("Add");
if (container.find("Add")) {
System.out.println("checked");
}
Container.Iterator it = container.iterator();
for (String item : container) {
System.out.println(item);
}
System.out.println("to String"+container.toString());
System.out.println(container.toArray().toString());
while (it.hasNext()) {
String curr = (String) it.next();
System.out.println(curr);
}
System.out.println("enter 1 string to manip");
String str=scanner.nextLine();
System.out.println("1-add string,2-find,3-sort,4-delete
all strings,5-display array,6-exit");
System.out.println("enter what you want to do");
int k=scanner.nextInt();
if(k==1){
container.add(str);
for (String item : container) {
System.out.println(item);
}
}
else if(k==2){
System.out.println(container.find(str));
}
else if(k==3) {
```

```
container.sort();
for (int i = 0; i < array.length; i++)
System.out.println(container.sort()[i]);
}
else if(k==4){
container.deleteAll();
}
else if(k==5) {
for (int i =0;i<array.length;i++)
System.out.println(array[i]);
}
else if(k==6){
System.out.println("Goodbye");
return;
}
}
}
```

Приклад використання програми



Висновок: Я набула навички розробки власних контейнерів, навчилась використовувати ітератори, ознайомилась з принципами серіалізації/десеріалізації об’єктів.