**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА  
ПОЛІТЕХНІКА”**

**Кафедра систем штучного інтелекту**

**Лабораторна робота №1  
з дисципліни  
«Об’єктно-орієнтоване програмування»**

**Виконав:  
студент групи КН-109**

**Качмар Олексій  
Викладач:  
Мочурад Л.І.**

**Львів – 2018 р.**

Лабораторна робота №2. Розробка власних контейнерів. Ітератори. Серіалізація/десеріалізація об’єктів. Бібліотека класів користувача

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 ​ ітерується​ для збереження початкових даних Вашого варіанту завдання з попередньої роботи ​ (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 ​ true​ , 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 Iterable​ . 3. В класі ітератора відповідно до ​ Interface Iterator​ реалізувати методи: ○ public boolean hasNext()​ ; ○ public String next()​ ; ○ public void remove()​ . 4. Продемонструвати роботу ітератора за допомогою циклів ​ while

​ и ​ for each

​ .

5. Забороняється використання контейнерів (колекцій) і алгоритмів з ​ Java Collections Framework​ . 6. Реалізувати і продемонструвати тривале зберігання/відновлення розробленого контейнера за допомогою ​ серіалізації/десеріалізації​ . 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.

Код програми:

**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.io.Serializable;

**import** java.util.Arrays;

**import** java.util.Iterator;

**import** java.util.Scanner;

**public** **class** MainClass

{

**public** **static** **void** main(String[] args) **throws** ClassNotFoundException

{

Scanner ob = **new** Scanner(System.***in***);

Container c = **new** Container();

Iterator<String> it = c.iterator();

String alphabet = "abcdefghigklmnopqrstuvwxyz";

System.***out***.println("Do an input\n");

String input = ob.nextLine();

String[] str =input.split(" ");

String s = "";

**for**(**int** word = 0;word <str.length;word++)

{

**for**(**int** ch = 0;ch<str[word].length();ch++)

{

**for**(**int** l = 0;l<alphabet.length();l++)

{

**if**(alphabet.charAt(l) == str[word].charAt(ch))

{

s+=Integer.*toString*(l+1);

s+=" ";

**break**;

}

}

System.***out***.print(str[word].charAt(ch));

System.***out***.print(" ");

}

}

System.***out***.println("\n" + s);

String fileName = "data.txt";

**while**(**true**)

{

System.***out***.println("\n1 - create container,add some words");

System.***out***.println("2 - show input,using iterator");

System.***out***.println("3 - show using methods from container");

System.***out***.println("4 - show work of serialization");

System.***out***.println("5 - show work of deserialization");

System.***out***.println("6 - exit");

**int** num = ob.nextInt();

**switch**(num)

{

**case** 1:

{

c.add("lexa");

c.add("prokach");

c.add("speede");

c.add("thinkable");

**break**;

}

//робота методів та ітератора

**case** 2:

{

**while**(it.hasNext())

{

System.***out***.println(it.next());

}

**break**;

}

**case** 3:

{

**if**(c.contains("speede"))

{

c.remove("speede");

System.***out***.println(c.size());

}

c.porivnjannja();

**for**(Object st : c.toArray())

{

System.***out***.println(st);

}

c.sort();

**for**(Object st : c.toArray())

{

System.***out***.println(st);

}

**break**;

}

//Serialization

**case** 4:

{

**try**

{

ObjectOutputStream os = **new** ObjectOutputStream(**new** FileOutputStream(fileName));

os.writeObject(c);

os.close();

}

**catch** (FileNotFoundException e)

{

e.printStackTrace();

}

**catch** (IOException e)

{

e.printStackTrace();

}

System.***out***.println("File was written in potik");

**break**;

}

**case** 5:

{

**try**

{

ObjectInputStream is = **new** ObjectInputStream(**new** FileInputStream(fileName));

Container c1 =(Container)is.readObject();

System.***out***.println("Words written in:" + c1);

is.close();

}

**catch** (FileNotFoundException e)

{

e.printStackTrace();

}

**catch** (IOException e)

{

e.printStackTrace();

}

**break**;

}

**case** 6:

{

System.***out***.println("System exited");

System.*exit*(0);

}

**default**:

System.***out***.println("unknown command.");

}

}

}

}

**class** Container **implements** Iterable<String> , Serializable

{

**private** Object [] string = **new** Object [] {};

**private** **int** counter;

**void** porivnjannja()

{

**int** Max = 0;

Object temp;

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

{

**for**(**int** word = 0;word<string.length - 1;word++)

{

**if**(((String)string[word+1]).length()>((String)string[word]).length())

{

temp = string[word];

string[word] = string[word+1];

string[word+1] = temp;

}

}

}

}

**void** sort()

{

Object temp;

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

{

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

{

**if** (((String)string[i]).compareTo((String)string[j])>0)

{

temp = string[i];

string[i] = string[j];

string[j] = temp;

}

}

}

}

**public** String toString()

{

String s = "";

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

{

s += string[i].toString();

s += " ";

}

**return** s;

}

**void** add(String s)

{

string = Arrays.*copyOf*(string, string.length + 1); //create new array from old array and allocate one more element

string[string.length - 1] = s;

}

**void** clear()

{

**this**.string = **new** Object[0];

}

**boolean** remove(String string)

{

counter = 0;

**int** length = **this**.string.length;

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

{

**if**(**this**.string[i] == string)

{

length--;

**int** start = i + 1;

**while**(start<**this**.string.length)

{

**this**.string[start - 1] = **this**.string[start];

start++;

}

**this**.string[**this**.string.length - 1] = "";

counter+=1;

**return** **true**;

}

}

**return** **false**;

}

Object[] toArray()

{

**return** **this**.string;

}

**int** size()

{

**return** string.length - counter;

}

**boolean** contains(String string)

{

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

{

**if**(**this**.string[i].equals(string))

{

**return** **true**;

}

}

**return** **false**;

}

**boolean** containsAll(Container container)

{

**for**(Object str : **this**.string)

{

**if**(!container.contains((String) str))

{

**return** **false**;

}

}

**return** **true**;

}

**public** Iterator<String> iterator()

{

**return** **new** MyIterator<String>();

}

**class** MyIterator<String> **implements** Iterator<String>

{

**int** indexInArr = 0;

**public** **void** remove()

{

**for**(**int** i = indexInArr + 1;i<string.length;i++)

{

string[i - 1] = string[i];

}

}

**public** **boolean** hasNext()

{

**if**(string.length >= indexInArr + 1)

{

**return** **true**;

}

**return** **false**;

}

**public** String next()

{

**if**(hasNext())

{

**return** (String)string[indexInArr++];

}

**return** **null**;

}

}

}

Висновок: Я набув навичок розробки власних контейнерів. Використав у роботі ітератор. Зрозумів як тривало зберігати та відновлювати стан об’єктів. ● Ознайомився з принципами серіалізації/десеріалізації об’єктів. Використав бібліотеки класів користувача.