Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Лабораторна робота №9

на тему:

«Розробка власних контейнерів. Ітератори.

Серіалізація/десеріалізація об’єктів.

Бібліотека класів користувача»

з курсу:

«ООП»

Виконав:
ст. гр. КН-110
Георгій Брусенцов
Прийняв:
Гасько Р.Т

Львів – 2018 р.

Мета роботи:

● *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.*

Розробник: дивитися вище, Варіант 1

Вимоги:

1. Розробити клас-контейнер, що ітерується (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 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 .*
http://docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html

3. В класі ітератора відповідно до Interface Iterator (http://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html) 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 - https://docs.oracle.com/javase/8/docs/technotes/guides/collections/

6. Реалізувати і продемонструвати тривале зберігання/відновлення розробленого контейнера за допомогою серіалізації/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.

Приклад коду:

```
package ua.lpnuai.oop.brusentsov09;

import java.io.*;
import java.util.Iterator;

public class NotArrayList implements Serializable {

    private String[] array = new String[10];
    private int index = 0;
    private int size = 10;

    public NotArrayList(String[] arr) {
        for (int i = 0; i < arr.length; i++) {
            if (enoughSpace(index + 1)) {
                this.add(arr[i]);
            } else {
                array = this.increaseSize(array);
                this.add(arr[i]);
            }
        }
```

```
}

public void add(String elem) {
   array[index++] = elem;
}

public void getList() {
   for (int i = 0; i < index; i++) {
      System.out.print(array[i] + " ");
   }
}

@Override
public String toString() {
   StringBuilder string = new StringBuilder("[");
   for (int i = 0; i < index; i++) {
      if (i == index - 1) {
         string.append(array[i]);
      } else string.append(array[i] + "," + " ");
   }
   string.append("]");
   return string.toString();
}

private boolean enoughSpace(int index) {
   return index <= size;
}

private String[] increaseSize(String[] array) {
   size = (size * 3) / 2 + 1;
   String[] arr_temp = new String[size];
   System.arraycopy(array, 0, arr_temp, 0, array.length);
   return arr_temp;
}

public void clear() {
   array = new String[size];
   index = 0;
}

public boolean remove(String string) {
   for (int i = 0; i < index; i++) {
      if (array[i].equals(string)) {
         String[] arr_temp = new String[size];
         System.arraycopy(array, 0, arr_temp, 0, i);
```

```
            System.arraycopy(array, i + 1, arr_temp, i, index - i);
            index--;
            array = arr_temp;
            return true;
        }
    }
    return false;
}

public boolean remove(int it){
    if (it <= index) {
        String[] arr_temp = new String[size];
        System.arraycopy(array, 0, arr_temp, 0, it);
        System.arraycopy(array, it + 1, arr_temp, it, index - it);
        index--;
        array = arr_temp;
        return true;
    }
    else return false;
}
public String[] toArray() {
    return array;
}

public int size() {
    return index;
}

public boolean contains(String string) {
    for (int i = 0; i < index; i++) {
            if (array[i].equals(string)){
                return true;
        }
    }
    return false;
}

public boolean containsAll(String[] strings){
    int found = 0;
    for (int i = 0; i < index; i++){
        for(int j = 0; j < strings.length; j++){
            if (array[i].equals(strings[j])){
                found++;
            }
        }
```

```
    }
    return found == strings.length;
}

public Iterator<String> iterator() {
    return new Itr();
}

public void save(Object object){
    try {
        FileOutputStream fos = new FileOutputStream("data.bin");
        ObjectOutputStream os = new ObjectOutputStream(fos);
        os.writeObject(object);
        os.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public NotArrayList load(){
    NotArrayList list = new NotArrayList(new String[]{""});
    try {
        FileInputStream fis = new FileInputStream("data.bin");
        ObjectInputStream ois = new ObjectInputStream(fis);

        list = (NotArrayList) ois.readObject();

        ois.close();

    } catch (IOException e) {
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
    return list;
}
private class Itr implements Iterator<String>{
    int next;
    int current = -1;

    Itr() {}

    public boolean hasNext(){
        return next != index;
    }
```

```
        public String next(){
            int i = next;
            next = i+1;
            String[] data = NotArrayList.this.array;
            return data[current = i];
        }

        public void remove(){
            NotArrayList.this.remove(current);
            next = 0;
            current = -1;
        }
    }
}
```

Висновки: *я навчився розробляти власні контейнери, ітератори. серіалізацію/десеріалізацію об'єктів та бібліотеку класів користувача.*