МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”
Кафедра систем штучного інтелекту

Лабораторна робота № 9

з дисципліни
«Об’єктно-орієнтоване програмування»

Виконала:
студентка групи КН-109
Пелещак Ю. М.
Викладач:
Гасько Р.Т.

Львів – 2018 р.

Лабораторна робота №9.
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 (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 рядківз 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.

Частина програми,що відповідає за виконання поставленої задачі:

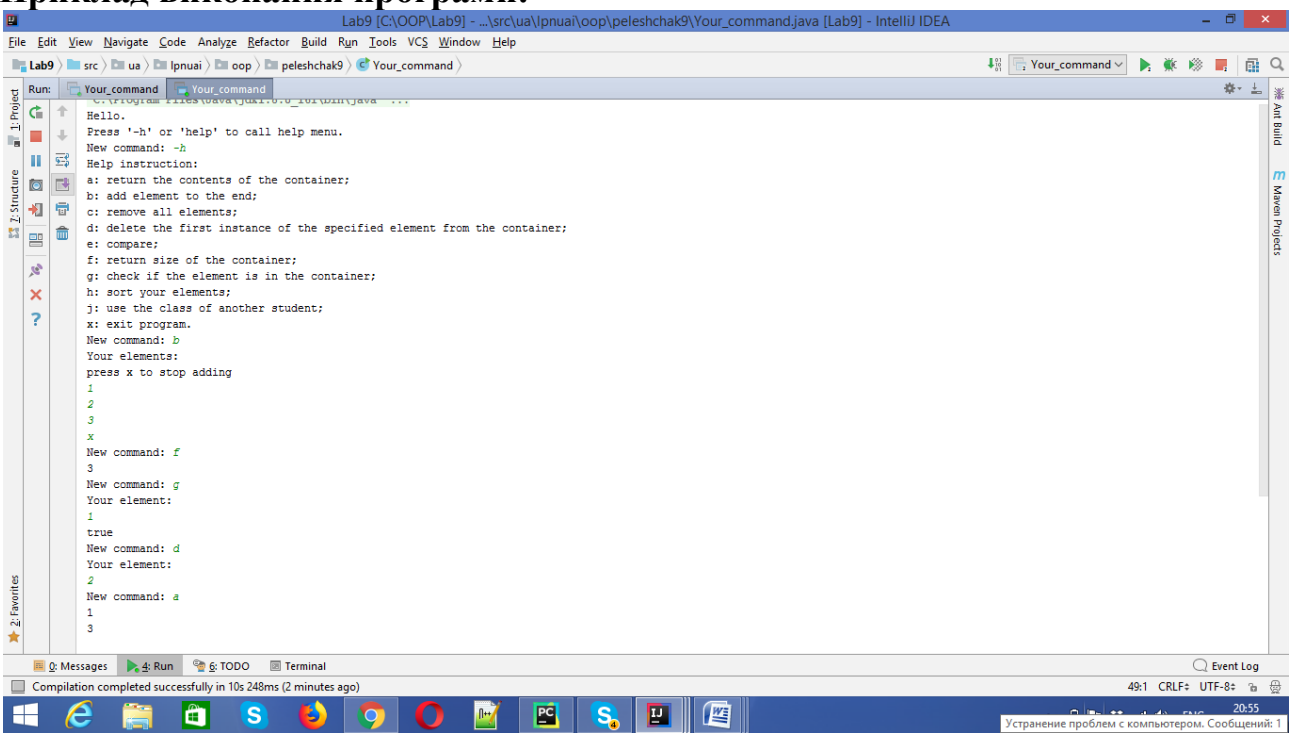
```
package
ua.lpnuai.oop.peleshchak9;
                                import java.io.*;
                                import java.util.Date;
                                import java.util.Iterator;
                                import java.util.NoSuchElementException;
                                import java.util.Scanner;
                                public class Container implements Containerr, Iterable<String>, Serializable{
                                    private static final long serialVersionUID = 1L;
                                    private int size = 0;
                                    private String[] data = null;
                                    public Container(){
                                        data = new String[0];
                                    }
                                    @Override
                                    public void add(String string) {
                                        if(data.length <= size)
                                            realloc();
                                        data[size++] = string;
                                    }
                                    public String get(int index){
                                        return index >= size ? null : data[index];
                                    }
                                    @Override
                                    public void clear() {
                                        size = 0;
                                        realloc();
                                    }
                                    @Override
                                    public boolean remove(String string) {
                                        for(int i = 0; i < size; ++i)
                                            if(data[i].equals(string)){
                                                for(int j = i; j < size - 1; ++j)
                                                    data[j] = data[j + 1];
                                                size--;
                                                return true;
                                            }
                                        return false;
                                    }
                                    @Override
```

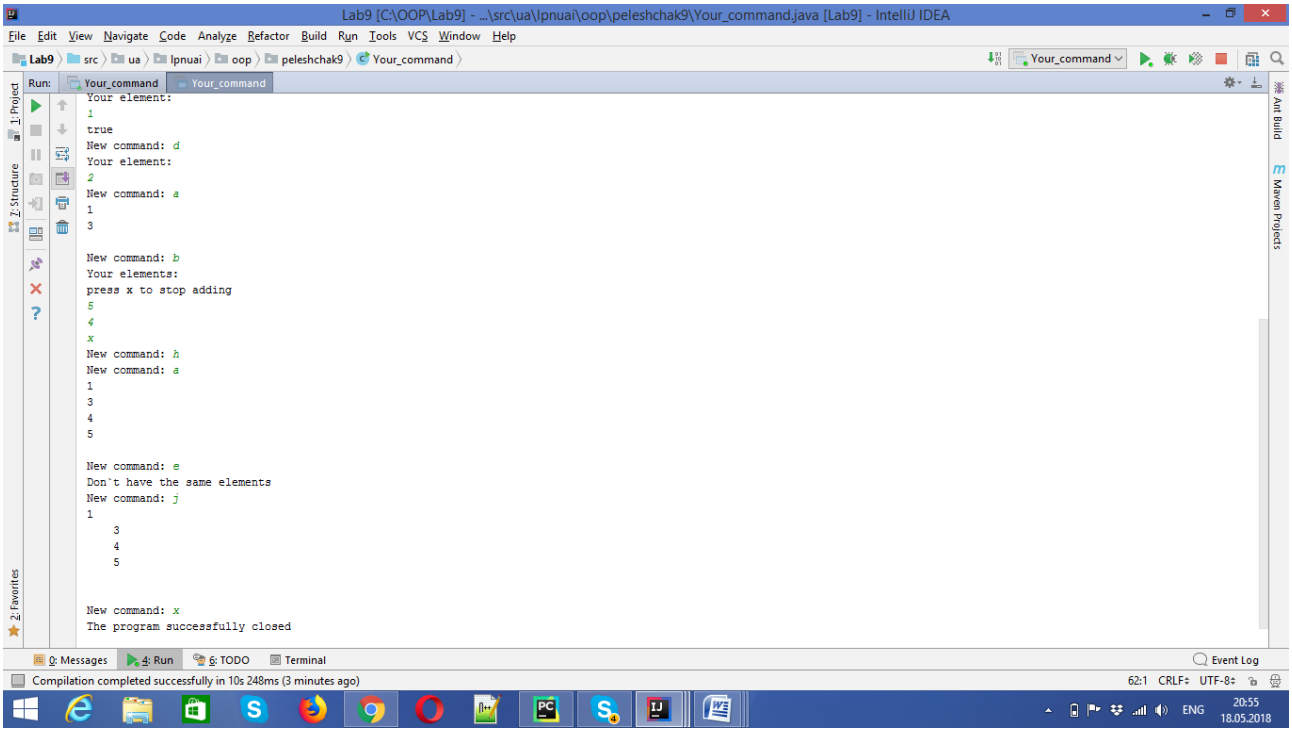
```
public Object[] toArray() {
    return data;
}
@Override
public int size() {
    return size;
}
@Override
public boolean contains(String string) {
    for(int i = 0; i < size; ++i)
        if(data[i].equals(string))
            return true;
    return false;
}
@Override
public boolean containsAll(Container container) {
    String[] s = (String[]) container.toArray();
    for(String s1 : s)
        if(!contains(s1))
            return false;
    return true;
}
@Override
public Iterator<String> iterator() {
    return new Itr<String>();
}
public String toString(){
    StringBuffer sb = new StringBuffer();
    for(Iterator<String> it = iterator(); it.hasNext();)
        sb.append(it.next() + "\n");
    return sb.toString();
}
public void sort(){
    qsort(0, size - 1);
}
private void qsort(int b, int e){
    String tmp;
    int l = b;
    int r = e;
    int m = (l + r) / 2;
    while(l <= r){
        while(data[l].compareTo(data[m]) < 0)l++;
        while(data[r].compareTo(data[m]) > 0)r--;
        if(l <= r){
            tmp = data[l];
            data[l] = data[r];
            data[r] = tmp;
            l++;
            r--;
        }
    }
    if(b < r)
        qsort(b, r);
    if(e > l)
        qsort(l, e);
```

```
    }
    @Override
    public void compare() {
        String equalElems = "";
        int countOfEqual = 0;
        for (int i = 0; i < data.length; ++i) {
            for (int j = i + 1; j < data.length; ++j) {
                if (data[i].equals(data[j])) {
                    equalElems += data[i];
                    countOfEqual++;
                }
            }
        }
        if (equalElems.isEmpty()) {
            System.out.println("Don`t have the same elements");
        } else {
            countOfEqual++;
            System.out.println(equalElems + " - " + countOfEqual);
        }
    }
    private void realloc(){
        String[] buf = data;
        data = new String[size == 0 ? 1 : size * 2];
        for(int i = 0; i < Math.min(buf.length, data.length); ++i)
            data[i] = buf[i];
    }
    public void serealization() {
        try {
            File f = new File("julia.dat");
            if(!f.exists())
                f.createNewFile();
            ObjectOutputStream o = new ObjectOutputStream(new
FileOutputStream(f));
            o.writeObject(this);
            o.flush();
            o.close();
        } catch (Exception e) {
            System.err.println("Error serializing object");
        }
    }
    public void deserealization() {
        try {
            File f = new File("julia.dat");
            if(!f.exists())
                f.createNewFile();
            ObjectInputStream o = new ObjectInputStream(new FileInputStream(f));
            Container buf = (Container)o.readObject();
            size = buf.size;
            data = buf.data;
            o.close();
        } catch (Exception e) {
            System.err.println("Error serializing object");
        }
    }
    public void read(){
```

```
            System.out.println("press x to stop adding ");
            Scanner in = new Scanner(System.in);
            String s;
            while(!(s = in.nextLine()).equals("x"))
                add(s);
        }
        class Itr<String> implements Iterator{
            int cursor = 0;
            int last = -1;
            @Override
            public boolean hasNext() {
                return cursor < size;
            }
            @Override
            public Object next() throws NoSuchElementException {
                return cursor == size ? null : data[cursor++];
            }
            @Override
            public void remove() {
                if(last > 0 && last < size)
                    Container.this.remove(data[last]);
            }
        }
    }
```

Приклад виконання програми:





Висновок: під час виконання лабораторної роботи №9 я створила програму відповідно до поставленої задачі.