МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота N2

З дисципліни

«Алгоритмізація та програмування»

Виконав:

Студент групи КН-108

Воробель Адріан

Викладач:

Грабовська Н.Р.

Львів – 2019р.

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.

1. 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.

1.1 Розробник

Воробель Адріан, КН-108, номер варіанту індивідуального завдання – 5.

1.2 Задача

Опис програми:

Клас-контейнер , що реалізує згадані вище методи , та клас-ітератор.

2.1 Засоби ООП

Використано три класи (Main, Conteiner, Iterator), імплементація інтерфейсів, наслідування.

2.2 Ієрархія та структура класів

Клас Conteiner – клас з основним функціоналом, реалізує різного виду методи.

Клас Iterator – клас, що імплементує методи інтерфейсу Interface Iterator.

Клас Main – реалізує методи з вище вказаних класів.

```java
package com.company;

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

public class Conteiner implements Iterable<String>, Serializable {

    protected static String [] lines = new String [0];
    protected static int pointer = 0;


    public String toString (){
        String line = new String();
        for (int i = 0; i < lines.length; i++){
            line += lines[i];
        }
        return line;
    }//ok

    void add(String addOne){
        lines = Arrays.copyOf(lines,  newLength: lines.length +1);
        lines[lines.length-1] = addOne;
    }//ok

    void clear(){
        lines = new String[0];
    }//ok

    boolean remove(String str){
        boolean removed = false;
        int j = 0;
        if (str != null){
            String [] buffer = new String[lines.length-1];
            for (int i=0; i< lines.length; i++) {
                if(lines[i].equals(str)) {
                    removed = true;
                    continue;
                }
                else {
                    buffer[j] = lines[i];
                    if (j!= lines.length-1)
                        j++;
                    continue;
                }
            lines = new String[0];
            lines = buffer;
        }
        else
            System.out.println("no such string");
        return removed;
    }//ok

    Object[] toArray(){
        String [] array =  Arrays.copyOf(lines, lines.length);
        return array;
    }//ok

    int size(){
        return lines.length;
    }//ok

    boolean contains(String check){
        boolean checked = false;
        for (int i = 0; i < lines.length; i++){
            if (lines[i].equals(check))
                checked = true;
            else
                continue;
        }
        return checked;
    }//ok

   boolean Serialize (String file){
       FileOutputStream fos;
       try {
           fos = new FileOutputStream(file);
       } catch (FileNotFoundException e) {
           System.out.println("file not found");
           return false;
       }
       try {
           ObjectOutputStream oos = new ObjectOutputStream(fos);
           oos.writeObject(lines);
           oos.flush();
           oos.close();
           fos.close();
           return true;
       }catch (IOException e) {
           System.out.println("fail with serialize");
```

```
            System.out.println("fail with serialize");
            return false;
        }
    }//ok

    boolean Deserialize (String file){
        try{
            FileInputStream fis = new FileInputStream(file);
            ObjectInputStream oin = new ObjectInputStream(fis);
            String[] buffer = (String[]) oin.readObject();
            for(String s: buffer){
                System.out.println(s);
            }
            return true;
        }catch(IOException e){
            System.out.println("fail with deserialize");
            return false;
        }catch (ClassNotFoundException e){
            System.out.println("fail with deserialize");
            return false;
        }
    }//ok

     @Override
     public Iterator<String> iterator() {
         return new com.company.Iterator();
     }

     public boolean compare(int a, int b){
         boolean same = false;
         if (lines[a] == lines[b])
             same = true;
         return same;
     }

     public void sort(){
        boolean sorted = false;

        while (!sorted){
         for(int i = 0; i< lines.length-1; i++){

             if (lines[i].charAt(0) == lines[i+1].charAt(0))
                 continue;
             else{
                    char left = lines[i].charAt(0);
                    char right = lines[i+1].charAt(0);
                   if (left < right)
                       continue;
                   else {
                       String buffer = lines[i];
                       lines[i] = lines[i+1];
                       lines[i+1] = buffer;
                       continue;
                   }
               }
        }
           boolean srtd = false;
           for(int i = 0; i< lines.length-1; i++){
               if (lines[i].charAt(0) < lines[i+1].charAt(0))
                   srtd = true;
               else {
                   srtd = false;
                   break;
               }
           }
        sorted = srtd;
        }
    } //ok

    public int search(String str){
        int index=lines.length +1;
        for (int i = 0; i< lines.length; i++){
            if (lines[i] == str)
                index = i;
            else
                continue;
        }
        return index;
    }//ok
}
```

```java
package com.company;

public class Iterator extends Conteiner implements java.util.Iterator {

    @Override
    public boolean hasNext() {
        boolean has = false;
        if ( pointer + 1 != lines.length+1 )
            has = true;
        return has;
    }
    @Override
    public String next() {
        String nextIter = lines[pointer++];
        return nextIter;
    }
    @Override
    public void remove(){
         lines[pointer] = null;
    }
}
```

3. Варіанти використання

Програма може використовуватись для зберігання стрічок та проведення махінацій з ними за допомогою визначених методів.

ВИСНОВКИ

У ході роботи розвинулись навички написання власних контейнерів , власних ітераторів, навички роботи із тривалим зберіганням даних(серіалізація та десеріалзація).