Тема: Розробка власних контейнерів. Ітератори.
Мета: Набуття навичок розробки власних контейнерів. Використання ітераторів.

1 ВИМОГИ

1.1 Розробник

Інформація про розробника:
- Когутенко Олександр Олексійович;
- КІТ-119Д;
- 11 варіант.

1.2 Загальне завдання

1. Розробити клас-контейнер, що ітерується для збереження початкових даних завдання л.р. №3 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.

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП

Використовується наслідування, інтерфейс, поліморфізм.

2.2 Ієрархія та структура класів

Використовую 6 классів: Array, ArrayIterator, HelperClass, InteractiveConsole, Main, SaveArray.
Array використовую як інтерфейс для класу контейнеру.
ArrayIterator використовую як особисту реалізацію ітератора.
HelperClass допоміжний класс для розрахунків.
InteractiveConsole клас для налагодженого спілкування програми з користувачем.
Main клас який має точку входу у програму.
SaveArray клас контейнер який має все необхідні методи маніпулятори.

2.3 Важливі фрагменти програми

```
package ua.khpi.oop.kogutenko05;


import java.util.Iterator;

/**
 * The interface Array.
 *
 * @param <E> the type parameter
 */
public interface Array<E> extends Iterable<E>{

    /**
     * повертае елемент за заданим індексом.
     *
     * @param index the index
     * @return e
     * @throws Exception the exception
     */
    E get(int index) throws Exception;

    /**
     * повертає вміст контейнера у вигляді рядка;
     *
     * @return
     */
    String toString();

    /**
```

```
    * додає вказаний елемент до кінця контейнеру;
    *
    * @param el the el
    */
   void add(E el);

   /**
    * видаляє всі елементи з контейнеру;
    */
   void clear();

   /**
    * видаляє перший випадок вказаного елемента з контейнера;
    *
    * @param index the el
    * @return
    */
   boolean remove(int index);

   /**
    * повертає масив, що містить всі елементи у контейнері;
    *
    * @return the e [ ]
    */
   E[] toArray();

   /**
    * повертає кількість елементів у контейнері;
    *
    * @return the int
    */
   int size();

   /**
    * повертає true, якщо контейнер містить вказаний елемент;
    *
    * @param el the el
    * @return the boolean
    */
   boolean contains(E el);

   /**
    * повертає true, якщо контейнер містить всі елементи з зазначеного у параметрах;
    *
```

```
     * @param el the el
     * @return the boolean
     */
    boolean containsAll(E[] el);

    /**
     * повертає ітератор відповідно до Interface Iterable.
     * @return
     */
    public Iterator<E> iterator();
}
```

```
package ua.khpi.oop.kogutenko05;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.ConcurrentModificationException;
import java.util.Iterator;
import java.util.NoSuchElementException;

/**
 * Итератор для массива
 *
 * @param <E> the type parameter
 */
public class ArrayIterator<E> implements Iterator<E>/*, Serializable,
Array<E>*/{
    private int index = 0;
    /**
     * The Values.
     */
    E[] values;

    /**
     * Instantiates a new Array iterator.
     *
     * @param values the values
     */
    ArrayIterator(E[] values)
    {
        this.values = values;
    }

    @Override
```

```
    public boolean hasNext() {
        return index < values.length;
    }

    @Override
    public E next() {
        return values[index++];
    }

    @Override
    public void remove() {
        throw new UnsupportedOperationException("Cannot remove item from
array.");
    }
}
```

```
package ua.khpi.oop.kogutenko05;

import java.util.Scanner;

/**
 * The type Interactive console.
 */
public class InteractiveConsole
{
    /**
     * Start console.
     */
    public void startConsole() {
        boolean check = true, checkHelpLine = true;
        String input, nikname;
        HelperClass helper = new HelperClass();
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.print("Input your nikname: ");
            nikname = scanner.nextLine();
            while (check) {
                if (checkHelpLine) {
                    System.out.println("Hello, my name is Alex Kogutenko\n"
                            + "I am from Ukrain and studing at NTU \"KHPI\"\n"
                            + "This is a test console-project with a debug programs.\n"
                            + "Such commands are present so far:\n"
                            + "\t-h | -help   \t-\t command for summary information about
other commands (important to remmember!)\n"
                            + "\t-d | -debug  \t-\t file debugger command\n"
```

```
                    + "\tchtext       \t-\t changed the text as in the past lab work (lab
work 3)\n"
                    + "\tshow         \t-\t show reserved array.\n"
                    + "\tedit        \t-\t edit reserved array.\n"
                    + "\texit        \t-\t exit form program\n");
            checkHelpLine = false;
        }
        System.out.print(nikname + "@" + nikname + ": ");
        input = scanner.nextLine();
        switch (input) {
            case " ": {
                break;
            }
            case "-h": {
                helper.printHelpInfo();
                break;
            }
            case "-help": {
                helper.printHelpInfo();
                break;
            }
            case "-d": {
                helper.printDebuggerInHelper();
                break;
            }
            case "-debug": {
                helper.printDebuggerInHelper();
                break;
            }
            case "chtext": {
                helper.changedText();
                break;
            }
            case "show": {
                helper.printSaves();
                break;
            }
            case "edit": {
                helper.edit();
                break;
            }
            case "exit": {
                check = false;
                break;
            }
```

```
                default: {
                    System.out.println("(" + input + ") I don't know this command :(");
                    break;
                }
            }
        }
        System.out.println("GOOD BEY!!!");
    } catch (Exception e) {
        System.out.println(e);
        check = false;
    }
  }
}
```

```
package ua.khpi.oop.kogutenko05;
import java.util.Iterator;


/**
 * The type Save array.
 *
 * @param <E> the type parameter
 */
public class SaveArray<E extends Object> implements Array<E>{

  private E[] arrayData;
  //private ;

  /**
   * Instantiates a new Save array.
   */
  public SaveArray()
  {
    arrayData = (E[]) new Object[0];
  }

  @Override
  public void add(E el) {
    try
    {
        E[] temp = arrayData;
        arrayData = (E[]) new Object[temp.length + 1];
        System.arraycopy(temp, 0 , arrayData, 0, temp.length);
        arrayData[arrayData.length - 1] = el;
```

```
    }
    catch (ClassCastException ex)
    {
        ex.printStackTrace();
    }
}

@Override
public void clear() {
    while (size() > 1) {
        remove(size()-1);
    }
    arrayData = (E[]) new Object[0];
}

private int findIndexOfElement(E el)
{
    int index = 0;
    if(size() > 0) {
        /*for(E elem : arrayData)
        {
            if(elem == el)
            {
                return index;
            }
            index++;
        }*/
        for(;index < size(); index++) {
            System.out.println("-" + arrayData[index] + "---" + el.toString() + "-");
            if(arrayData[index] == el) {
                return index;
            }
        }
        return -1;
    }
    else if (size() == 0) {
        return index;
    }
    else {
        return -1;
    }
}

@Override
public boolean remove(int index) {
```

```
        //int index = findIndexOfElement(el);
        try {
            if (index == 0 && size() > 1) {
                E[] temp = arrayData;
                arrayData = (E[]) new Object[temp.length - 1];
                System.arraycopy(temp, 1, arrayData, 0, temp.length - 1);
                return true;
            }
            else if (index == 0 && size() == 1) {
                arrayData = (E[]) new Object[0];
                return true;
            }
            else if (index > 0 && size() == 0) {
                return false;
            }
            else if (index > 0 && size() > 0) {
                E[] temp = arrayData;
                arrayData = (E[]) new Object[temp.length - 1];
                System.arraycopy(temp, 0, arrayData, 0, index);
                int amountElemAfterIndex = temp.length - index - 1;
                System.arraycopy(temp, index + 1, arrayData, index,
amountElemAfterIndex);
                return true;
            }
            else {
                return false;
            }
        }
        catch(ClassCastException ex){
            ex.printStackTrace();
        }
        return false;
    }

    @Override
    public E[] toArray() {
        return null;
    }

    @Override
    public int size() {
        return arrayData.length;
    }

    @Override
```

```
public boolean contains(E elem) {
    if(size() == 0)
    {
        return arrayData[0] == elem;
    }else if(size() > 0)
    {
        for(E el : arrayData)
        {
            if(el == elem) return true;
        }
        return false;
    }
    return false;
}

private int sumInteger(int[] arr)
{
    int sum = 0;
    for(int i = 0;i < arr.length;sum+=arr[i++]);
    return sum;
}
@Override
public boolean containsAll(E[] arr) {
    if(size() == 0 && arr.length == 0)
    {
        return arrayData[0] == arr[0];
    }else if(size() > 0 && arr.length == 0)
    {
        for(E el : arrayData)
        {
            if(el == arr[0]) return true;
        }
        return false;
    }else if(size() > 0 && arr.length > 0)
    {
        int check[] = new int[arr.length];
        int lenCheck = arr.length - 1;
        //for(int i = 0, k = 0; i < lenCheck; check[i] = k, i++);
        try
        {
            for(E el : arr)
            {
                check[lenCheck--] = contains(el) ? 1 : 0;
            }
            if(sumInteger(check) == arr.length)
```

```
            {
                return true;
            } else {
                return false;
            }
        }catch(ArrayIndexOutOfBoundsException ex)
        {
            ex.printStackTrace();
        }

    }else if(size() == 0 && arr.length > 0)
    {
        for(E el : arr)
        {
            if(el == arrayData[0]) return true;
        }
        return false;
    }
    return false;
}

@Override
public String toString()
{
    String out = "size of reserved array is " + size() + "\n Content:\n";
    if(size() == 0)
    {
        out += "Array is empty";
    }else{
        int i = 1;
        for(E el : arrayData)
        {
            out = out + i++ + " : " + (String)el + "\n";
        }
    }
    return out;
}

public E get(int index){
    try
    {
        if(index < size())
            return arrayData[index - 1];
        else if (index < 0 || index > size()) {
            throw new Exception("Out of range!!!");
```

```
        }
    }catch(Exception ex)
    {
        ex.printStackTrace();
    }
    return null;
}


@Override
public Iterator<E> iterator() {
    return new ArrayIterator<>(arrayData);
}
}
```

3 ВАРІАНТИ ВИКОРИСТАННЯ

Програма предназначена за для розширеного використання лабороторної роботи №3 за допомогою використання параметрів перед запуском та збереженням всіх речень які змінювались.

```
Input your nikname: alex
Hello, my name is Alex Kogutenko
I am from Ukrain and studing at NTU "KHPI"
This is a test console-project with a debug programs.
Such commands are present so far:
    -h | -help      -    command for summary information about other commands (important to remmember!)
    -d | -debug     -    file debugger command
    chtext          -    changed the text as in the past lab work (lab work 3)
    show            -    show reserved array.
    edit            -    edit reserved array.
    exit            -    exit form program
```

Рисунок 5.1 - початок роботи

```
alex@alex: show
size of reserved array is 0
 Content:
Array is empty
alex@alex: -d

Date: 18.12.2020
Time: 00:03:59
values of the variables:
StringBuilder 'strBefore' has 'EmptyLine'
String 'strAfter' has 'EmptyLine'
Integer 'count' has '0'
Integer 'length' has '0'
alex@alex: chtext
```

Рисунок 5.2 - робота show та -d

```
alex@alex: chtext

Enter the text. In the text,
replace the words of the specified length with the specified line
Main line: low is empty
Enter number of letters in word which you want to changed: 2
Enter word to replace: +++
-------------------------------------------
No changed line: low is empty
Result:          low +++ empty
alex@alex: show
size of reserved array is 1
 Content:
1 : low is empty
```

Рисунок 5.3 - робота chtext та та show після запису

```
Such commands are present so far:
    remove          -    remove element
    contain         -    boolean contain of some var
    clear           -    clear all array
    exit            -    exit from edit
-> contain
Enter some string which can contains at this array
-> adwd
we don't search this string
```

Рисунок 5.3 - робота з edit container

ВИСНОВКИ

Набув навичок розробки власних контейнерів та використовування ітераторів.