

## Звіт

### Лабораторна робота №5

## РОЗРОБКА ВЛАСНИХ КОНТЕЙНЕРІВ. ІТЕРАТОРИ

**Мета:** Набуття навичок розробки власних контейнерів. Використання ітераторів.

### ВИМОГИ

#### Розробник:

- Чугунов Вадим Юрійович;
- КІТ-119а;
- Варіант №24.

#### 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`.

## ОПИС ПРОГРАМИ

### Опис змінних

`Kontainer kontain;`            `// об'єкт створеного класу Kontainer`  
`IteratorMine iterator;`       `// об'єкт створеного класу IteratorMine`

### Ієрархія та структура класів

**Class** `Main` – точка входу в програму;

**Class** `IteratorMine` – inner-клас класу `Kontainer`;

**Class** `Kontainer` – розроблений клас-контейнер;

**Class** `TestHelper` – утилітарний клас, який має в собі `Split` метод;

**Class** `Function` - допоміжний клас в якому реалізуються функції.

## ТЕКСТ ПРОГРАМИ

### Текст файлу `Main.java`

```
package ua.oop.khpi.chugunov05;

import ua.oop.khpi.chugunov05.Kontainer.IteratorMine;

public class Main {

    /**
     * An entry point of program
     *
     * @param args
     */
}
```

```

public static void main(final String[] args) {

    // Container

    Kontainer kontain = new Kontainer();

    String str1;

    //Initial data of lab. 3

    str1 = Functions.initializeStr();

    System.out.println("-----Initial data of lab #3-----");

    Functions.showString(str1);

    System.out.println("\n-----");

    String[] array = TestHelper.SplitString(str1);

    System.out.println("\nData after work of one helper method:");

    Functions.showStringArray(array);

    kontain.aDDBack(str1);

    kontain.addElemOfArray(array);

    System.out.println("=====Container=====");

    System.out.println("Container contents:");

    kontain.showArray();

    System.out.println("\n\n\n");

    System.out.println("Using container method - to string():");

    System.out.println(kontain.arrayToStr());

    System.out.print("\nWrite with iterator: ");

    IteratorMine iterator = (Kontainer.IteratorMine) kontain.iterator();

    for (String s : kontain) {

        System.out.println(s);

    }

    iterator.next();

    iterator.remove();

    System.out.println();

    kontain.showArray();
}

```

```

        System.out.println("Size array: " + kontain.getSize());

        while (iterator.hasNext()) {

            System.out.println(iterator.next() + " ");

        }

        System.out.println("Clear array: ");

        kontain.clearArray();

        kontain.showArray();

    }

}

```

## Текст файлу Kontainer.java

```

package ua.oop.khpi.chugunov05;

import java.util.Arrays;
import java.util.Iterator;
import java.util.NoSuchElementException;

public class Kontainer implements Iterable <String> {

    /**
     * First size for array.
     */
    private final int size = 5;

    /**
     * Array contains all data.
     */
    private String[] array = new String[size];

    /**
     * Counter of number elements.
     */
    private int count = 0;

    /**
     * Showing array`s data.
     */
    void showArray() {

```

```

        if (count == 0) {
            System.out.println("Empty mass");
        } else {
            System.out.println();
            for (int i = 0; i < count; i++) {
                System.out.println(array[i]);
            }
        }
    }

/**
 * First size for array.
 *
 * @param str1 -
 */
void aDDBack(final String str1) {
    if (count == array.length) {
        array = Arrays.copyOf(array, array.length * 2);
        array[count++] = str1;
    } else {
        array[count++] = str1;
    }
}

/**
 * The override to add method for adding an elem of string array.
 *
 * @param str - string array
 */
public void addElemOfArray(final String[] str) {
    for (String i : str) {
        this.aDDBack(i);
    }
}

String arrayToStr() {
    StringBuilder str1 = new StringBuilder("");

```

```

        if (count != 0) {
            str1 = new StringBuilder(array[0]);
            str1.append(" ");
            for (int i = 1; i < count; i++) {
                str1.append(array[i]);
                str1.append(" ");
            }
        }
        return str1.toString();
    }
}

```

```

void clearArray() {
    array = null;
    count = 0;
}

```

```

int getSize() {
    return count;
}

```

```

boolean contains(final String str) {
    boolean cont = false;
    for (int i = 0; i < count; i++) {
        if (cont) {
            return cont;
        } else {
            String str1;
            str1 = array[i];
            cont = str.equals(str1);
            if (i == count - 1) {
                return cont;
            }
        }
    }
}
}

```

```

        return cont;
    }

    boolean remove(final String str) {
        boolean remov = false;
        int coun = 0;
        for (int i = 0; i < count; i++) {
            if (remov) {
                break;
            } else {
                remov = str.equals(array[i]);
                coun++;
                if (i == count - 1) {
                    break;
                }
            }
        }
        array[coun - 1] = null;
        for (int i = 0; i < count; i++) {
            array[coun - 1] = array[coun++];
        }
        count--;
        return remov;
    }

    public Object[] toArray() {
        if (array == null) {
            return null;
        }
        return Arrays.copyOf(array, count);
    }

    String elementByIndex(final int index) {
        return array[index];
    }

```

```

    }

    boolean containsAll(final Kontainer container) {
        boolean result;
        if (container.getSize() != count) {
            return false;
        }
        for (int i = 0; i < count; i++) {
            result = array[i].equals(container.elementAt(i));
            if (!result) {
                return false;
            }
        }
        return true;
    }

    @Override
    public Iterator<String> iterator() {
        return new IteratorMine();
    }

    public class IteratorMine implements Iterator<String> {
        /**
         * First size for array.
         */
        private int position = 0;

        @Override
        public boolean hasNext() {
            return position < count;
        }

        @Override
        public String next() {
            if (this.hasNext()) {
                return array[position++];
            }
        }
    }

```



```

        } else {
            throw new NoSuchElementException();
        }
    }

    @Override
    public void remove() {
        int temp = position;
        for (int i = position; i < count; i++) {
            array[temp++] = array[i + 1];
        }
        count--;
    }
}

```

## Текст файлу TestHelper.java

```

package ua.oop.khpi.chugunov05;

import java.util.ArrayList;
import java.util.List;

public class TestHelper {

    public static String[] SplitString(String text) {
        List<String> words = new ArrayList<>();
        StringBuilder builder = new StringBuilder();
        int count = 0;
        char symbol;
        while (count < text.length()) {
            for (int i = count; i < text.toCharArray().length; i++)
            {
                symbol = text.toCharArray()[i];
                if((int)symbol == 32 | (int)symbol == 33 |(int)symbol == 58|(int)symbol ==
44|(int)symbol == 46) {
                    count++;

```

```

        break;
    }
    builder.append(symbol);
    count++;
}
words.add(builder.toString());
builder = new StringBuilder();
}

if(builder.length() != 0) {
    words.add(builder.toString());
}
for (int i = 0; i < words.size(); i++) {
    if(words.get(i).length() == 0) {
        words.remove(i);
    }
}
String[] output = new String[words.size()];
for (int i = 0; i < words.size(); i++) {
    output[i] = words.get(i);
}
return output;
}
}

```

## Текст файлу Functions.java

```

package ua.oop.khpi.chugunov05;

import java.util.*;

public class Functions {

    static void showString(final String str) {
        System.out.print(str);
    }
}

```

```
}
```

```
static void showStringArray(final String[] arr){  
    for (int i = 0; i < arr.length ; i++) {  
        System.out.println(arr[i]);  
    }  
}
```

```
private static String[] differentWords(final String str) {  
    int w = 0;  
    int begin = 0;  
    int count = 0;  
    for (int i = 0; i < str.length(); i++) {  
        if (str.charAt(i) == ' ') {  
            count++;  
        }  
    }  
    String[] wordArr = new String[++count];  
    for (int i = 0; i < str.length(); i++) {  
        if (str.charAt(i) == ' ') {  
            wordArr[w] = str.substring(begin, i);  
            w++;  
            begin = i + 1;  
        }  
    }  
    wordArr[w] = str.substring(begin, str.length());  
    return wordArr;  
}
```

```
public static String initializeStr() {  
    Scanner scan = new Scanner(System.in);  
    System.out.print("Enter string: ");  
    String str1;  
    str1 = scan.nextLine();  
}
```

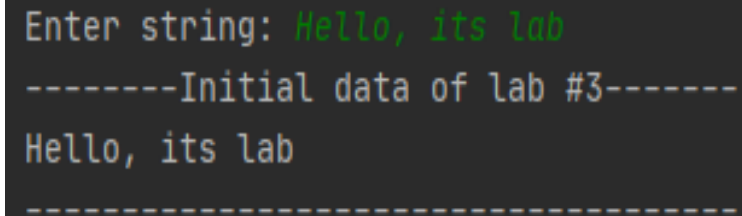
```

        return str1;
    }

    private static String reload(final String[] str2) {
        StringBuilder str1 = new StringBuilder("");
        str1 = new StringBuilder(str2[0]);
        str1.append(" ");
        for (int i = 1; i < str2.length; i++) {
            str1.append(str2[i]);
            str1.append(" ");
        }
        str1.deleteCharAt(str1.length() - 1);
        return str1.toString();
    }
}

```

## ВАРІАНТИ ВИКОРИСТАННЯ



```

Enter string: Hello, its lab
-----Initial data of lab #3-----
Hello, its lab
-----

```

a)

```
Data after work of one helper method:  
Hello  
its  
lab  
=====Container=====  
Container contents:  
  
Hello, its lab  
Hello  
its  
lab
```

б)

```
Using container method - to string():  
Hello, its lab Hello its lab  
  
Write with iterator: Hello, its lab  
Hello  
its  
lab  
  
Hello, its lab  
its  
lab  
Size array: 3  
its  
lab  
Clear array:  
Empty mass  
  
Process finished with exit code 0
```

в)

Рисунок 1 – Результат роботи програми

Дану програму можна використовувати для пошуку потрібного нам слова в тексті, та знаходження кількості повторів його в тексті. Використовувати контейнер для об'єктів та ітерування.

## ВИСНОВОК

Під час виконання лабораторної роботи було створено програму, яка має власний клас контейнер та ітератор. Реалізували та продемонстрували відповідні методи класу контейнера та ітератора.