Міністерство освіти і науки України

Національний технічний університет

«Харківський політехнічний інститут»

Кафедра «Обчислювальна техніка та програмування»

**ЗВІТ**

Про виконання лабораторної роботи № 5

«[Розробка власних контейнерів. Ітератори](https://oop-khpi.gitlab.io/#tasks)»

Керівник: Жилін Володимир Анатолійович

Виконавець: студентка гр. КІТ-320 Шпилька Д. О.

Харків 2021

Тема роботи: [Розробка власних контейнерів. Ітератори](https://oop-khpi.gitlab.io/#tasks)

Мета:

* Набуття навичок розробки власних контейнерів.
* Використання ітераторів.

Вимоги:

1. Розробити клас-контейнер, що [ітерується](http://docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html) для збереження початкових даних [завдання л.р. №3](https://oop-khpi.gitlab.io/%D0%B7%D0%B0%D0%B2%D0%B4%D0%B0%D0%BD%D0%BD%D1%8F/task03/#_4) 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).

1. 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().

1. Продемонструвати роботу ітератора за допомогою циклів *while* и *for each*.
2. Забороняється використання контейнерів (колекцій) і алгоритмів з [Java Collections Framework](https://docs.oracle.com/javase/8/docs/technotes/guides/collections/).

Код:

package ua.khpi.oop.Shpylka05;

import java.io.\*;

import java.util.\*;

public class main {

public static void main(String[] args) {

class Helper{

static String remover(String S){

System.out.println("Введите текст");

Scanner in = new Scanner(System.in);//Создание контейнера для ввода

String string = in.nextLine();//Ввод

String[] words = string.split("\\s+");//Разбитие предложения на массив слов

HashMap<String, Integer> wordToCount = new HashMap<>();//позволяет искать элемменты в строке

for (String word : words)

{

if (!wordToCount.containsKey(word))// запрашивает ключ(слово стринг)

{

wordToCount.put(word, 0);// определяет отдельные слова

}

wordToCount.put(word, wordToCount.get(word) + 1);//ищет совпадения слов

}

for (String word : wordToCount.keySet())//возвращает найденые совпадения слов

{

System.out.println(word + " " + wordToCount.get(word));

}

return string;

}

}

class Container implements Iterable<String> {

String[] our\_strings = {};

String s;

@Override

public Iterator<String> iterator(){

Iterator<String> it = new Iterator<String>() {

private int currentIndex = 0;

@Override

public boolean hasNext() {

return currentIndex < our\_strings.length && our\_strings[currentIndex] != null;

}

@Override

public String next() {

return our\_strings[currentIndex++];

}

@Override

public void remove() {

throw new UnsupportedOperationException();

}

};

return it;

}

String to\_string(){

StringBuilder temp\_buffer = new StringBuilder();

for (String our\_string : our\_strings) {

temp\_buffer.append(our\_string);

temp\_buffer.append(',');

}

return temp\_buffer.toString();

}

void add(String string){

s = string;

int i;

int length = our\_strings.length;

String[] new\_arr = new String[length + 1];

for (i = 0; i < length; i++){

new\_arr[i] = our\_strings[i];

}

new\_arr[length] = string;

our\_strings = new\_arr;

}

void clear(){

our\_strings = new String[0];

}

Object[] to\_array(){

Object[] resulting\_object = new Object[our\_strings.length];

int i = 0;

for (String S : our\_strings){

resulting\_object[i] = S;

i++;

}

return resulting\_object;

}

int size(){

return our\_strings.length + 1;

}

boolean contains(String string){

for (String S : our\_strings){

if (Objects.equals(S, string)){

return true;

}

}

return false;

}

boolean contains\_all(Container container){

return this == container;

}

boolean remove(String string) {

int length = our\_strings.length;

String[] new\_arr = new String[length - 1];

for (int i = 0, k = 0; i < length; i++){

if (our\_strings[i].equals(string)) {

continue;

}

new\_arr[k++] = our\_strings[i];

}

our\_strings = new\_arr;

return true;

}

void remover\_all(){

int i;

int length = our\_strings.length;

String[] new\_arr = new String[length];

for (i = 0; i < length; i++){

our\_strings[i] = Helper.remover(our\_strings[i]);

new\_arr[i] = our\_strings[i];

}

our\_strings = new\_arr;

}

void remover\_index(int index){

int i;

int length = our\_strings.length;

String[] new\_arr = new String[length];

for (i = 0; i < length; i++){

if (i == index){

our\_strings[i] = Helper.remover(our\_strings[i]);

}

new\_arr[i] = our\_strings[i];

}

our\_strings = new\_arr;

}

void sort(){

for(int i = 0; i < this.size(); i++) {

for (int j = i + 1; j < this.size(); j++){

if(our\_strings[i].toLowerCase().compareTo(our\_strings[j].toLowerCase()) > 0){

String temp = our\_strings[i];

our\_strings[i] = our\_strings[j];

our\_strings[j] = temp;

}

}

}

}

boolean compare\_strings(int index1, int index2){

return our\_strings[index1].equals(our\_strings[index2]);

}

}

}

}

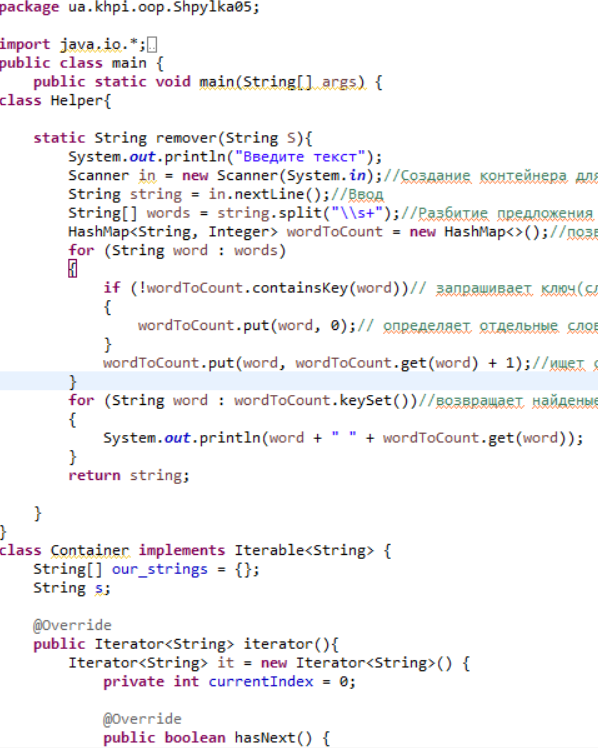


рис.1



рис.2

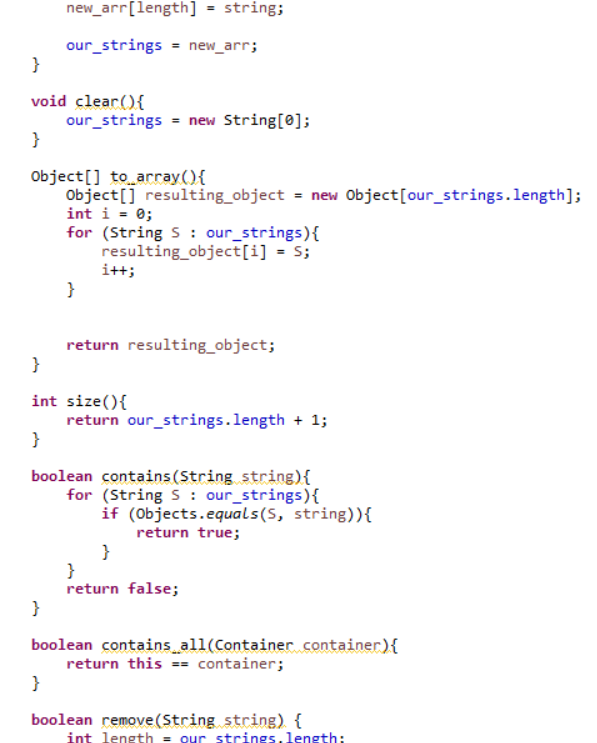


рис.3

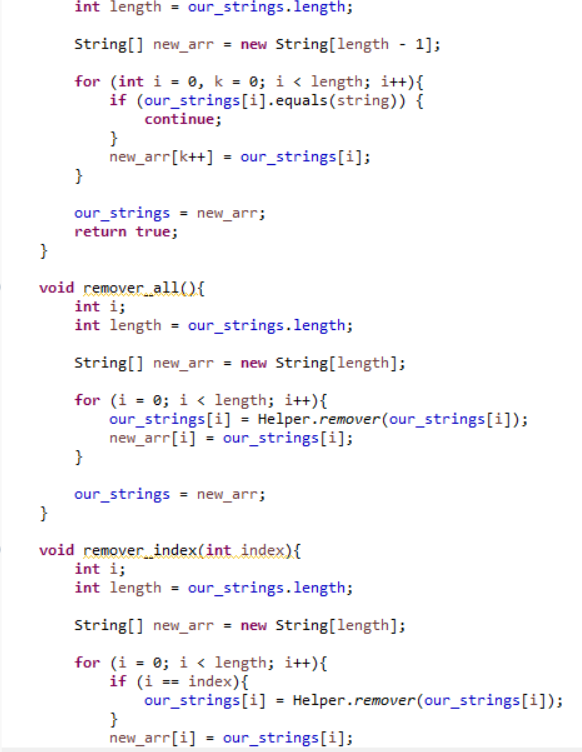


рис.4

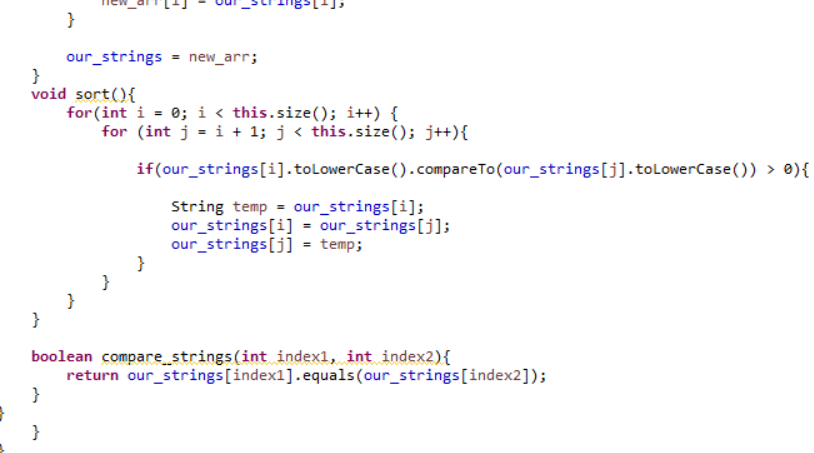


рис.5

Висновок: Було набуло навичок розробки власних контейнерів, використання ітераторів.