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

Мета:

* Набуття навичок розробки власних контейнерів.
* Використання ітераторів.

**1 ВИМОГИ**

**1.1 Розробник**

Інформація про розробника:

- Резніченко Дмитро Володимирович

- КІТ-319;

- 6 варіант.

**1.2 Загальне завдання**

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).
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/).

1.3 Задача

Поновити л.р.3

2 ОПИС ПРОГРАМИ

2.1 Засоби ООПСтворення власного класу контейнеру для зберігання даних підчас роботи. Для коректної роботи   
2.2 Ієрархія та структура класівКлас “Helper ” виконує роль допоміжного класу який виконує неосновні завдання наприклад : виведення результату або перевірка символів на відповідність. Клас-контейнер «Сontainer» зберігає всі дані в масиві та надає доступ до даних . Методи класу : додавання , видалення , пошук, кількість елементів.

Ітератор «Iterator» - засобіб послідовного доступу до вмісту контейнера; він є інтелектуальним вказівником, що «знає» як отримати доступ до елементів контейнера;

2.3 Важливі фрагменти програми

**public** **class** Сontainer {

**int** size=0;

String[] m\_data=**new** String[255];

**public** String toString() //повертає вміст контейнера у вигляді рядка

{

**if** (size ==-1) {

System.***out***.print("\nМасив пустий. Елементів немає. Попернуто null");

**return** **null**;

}

String temp=**new** String();

**for** (**int** i=0;i<size;i++)

temp+=m\_data[i];

**return** temp;

}

**void** add(String string) //додає вказаний елемент до кінця контейнеру;

{

**if** (size + 1 >= 255) **return**;

//m\_data[size++]=string;

String temp=**new** String();

**for** (**int** i=0;i<string.length();i++)

{

**if**((**char**)string.charAt(i)!=32)

temp+=string.charAt(i);

**else**{

m\_data[size++]=temp+" ";

temp=**new** String();

}

}

m\_data[size++]=temp+" ";

}

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

{

**while** (size!=0)

iterator().remove();

}

**boolean** remove (String string)// видаляє перший випадок вказаного елемента з контейнера;

{

**if** (size ==0) **return** **false**;

**for** (**int** i=0;i<size;i++)

**if** (m\_data[i]==string)

{ **for** (; i < size-1; i++)

m\_data[i]=m\_data[i+1];

**this**.m\_data[--size] = **null**;

**return** **true**;

}

**return** **false**;

}

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

{

**return** size;

}

**boolean** contains(String string)/// повертає true, якщо контейнер містить вказаний елемент;

{

**if** (size ==0) **return** **false**;

**for** (**int** i=0;i<size;i++)

**if** (m\_data[i]==string)

**return** **true**;

**return** **false**;

}

Object[] toArray() ///повертає масив, що містить всі елементи у контейнері;

{

**return** m\_data;

}

**boolean** containsAll(Сontainer container)/// повертає true, якщо контейнер містить всі елементи з зазначеного у параметрах;

{

**if**(container.size==size)

**if**(container.m\_data==m\_data)

**return** **true**;

**return** **false**;

}

**public** Iterator<String> iterator() ///повертає ітератор відповідно до Interface Iterable.

{

**return** **new** m\_Iterator();

}

**public** **class** m\_Iterator **implements** Iterator<String>

{

**int** index = 0;

**public** **boolean** hasNext() {

**if**(index<size)

**return** **true**;

**return** **false**;

}

**public** String next()

{

**return** m\_data [index++];

}

**public** **void** remove()

{

**for** (**int** i=index; i < size-1; i++)

m\_data[i]=m\_data[i+1];

m\_data[--size] = **null**;

//throw new UnsupportedOperationException("remove");

}

}

}

////////////////////////////////////////////////

/\*\*

\* \brief L3-Help

\* \author Dmitry Reznichenko

\* \version 1.0

\* \date septemer 2020 года

\* \warning Данный класс создан тільки для начальних цілей да робти основного класу main

\*

\* Завдання класу:

\* Виконувати функції які потріні щоб виконувати основне завдання

\*/

**public** **class** Helper

{

//static functions

**public** **static** **boolean** СonditionalСheck(**char** text)/// перевіряє символ на умову

{

**return** ( text>64&&text<91)||( text>96&&text<=123)||text==32||text==44;

}

**public** **static** **void** PrintLine(String text)///вивод тексту на консоль

{

System.***out***.println("\n\nВаш текст(дублювани): "+text);

}

**public** **static** **void** PrintNewLine(String text)///вивод тексту на консоль із надписом відредагований

{

System.***out***.println("\n\nВаш текст(Відредагований): "+text);

}

**public** **static** Сontainer Task6 (Сontainer text)//////редагування тексту

{

String temp=**new** String();

String s=**new** String(text.toString());

**boolean** spaise=**false**;

**for** (**int** i=0;i<s.length();i++)

{

**if** (Helper.*СonditionalСheck*(s.charAt(i)))

{

**if** (s.charAt(i)==32) spaise=**true**;//перевірка на пробіл

**else**

{

**if**(s.charAt(i)==44)///перевірка на кому

{

temp=temp+s.charAt(i) ;

spaise=**true**;

}

**else** **if**(spaise==**true**)///написати пробіл

{ temp =temp+" "+s.charAt(i);

spaise=**false**;

}

**else** temp =temp+s.charAt(i);///написати символ

}

}

} text.clear();

text.add(temp);

**return** text;

}

}

//////////////////////////////////////////////

/\*\*

\* \brief L5

\* \author Dmitry Reznichenko

\* \version 1.0

\* \date septemer 2020 года

\* \warning Данный класс создан тільки для начальних цілей

\*

\* Завдання:

\*Ввести текст. З тексту видалити всі символи, крім пропусків, які не є буквами.

\* Пропуски, що повторюються, замінити на одиночні. Між послідовностями літер,

\* де знаходяться розділові знаки, залишити хоча б один пропуск ("a,b,c" -> "a, b, c").

\* Вивести початковий текст та результат.

\*/

import java.util.Scanner;

public class Reznichenko\_Dmitry\_05 {

public static Scanner in = new Scanner(System.in);

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

System.out.print("Введіть ваш текст: ");

Сontainer s=new Сontainer();

String s2=new String(in.nextLine());

s.add(s2);

Iterator<String> iterator=s.iterator();

Helper.PrintLine(s.toString());///вивод тексту

s=Helper.Task6(s);///основне завдання

Helper.PrintNewLine(s.toString());///;вивод відредагованого текту

System.out.print("\n Вивод через iтератор(while):");

while (iterator.hasNext())

{

System.out.print(iterator.next().toString());

}

}

}

**3 ВАРІАНТИ ВИКОРИСТАННЯ**

Після введення даних данні записуються у контейнер за допомогою команди add. Після в функцію редагування передаються дані контейнеру за допомогою команди toString(). Відредаговані данні перезаписуються у контейнер. Результат виводиться на екран звичайним способом та через ітератор.

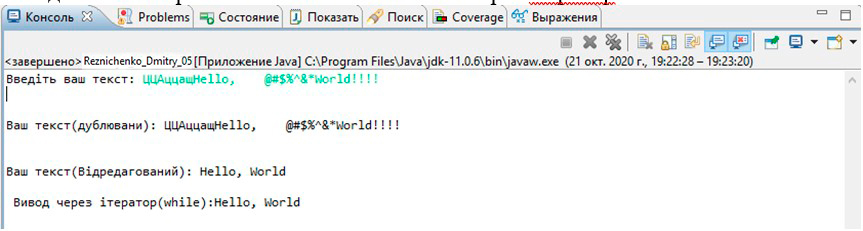


Рисунок 1 – результат редагування тексту

ВИСНОВКИ

Під час виконання лабораторної роботи було набуто навички з розробки власних контейнерів та створення і використання ітераторів.