**Звіт**

# Лабораторна работа 5.

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

**Мета роботи**:

* Набуття навичок розробки власних контейнерів.
* Використання ітераторів.

1. **ВИМОГИ**
2. 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.
3. В контейнері реалізувати та продемонструвати наступні методи:
   1. String toString() повертає вміст контейнера у вигляді рядка;
   2. void add(String string) додає вказаний елемент до кінця контейнеру;
   3. void clear() видаляє всі елементи з контейнеру;
   4. boolean remove(String string) видаляє перший випадок вказаного елемента з контейнера;
   5. Object[] toArray() повертає масив, що містить всі елементи у контейнері;
   6. int size() повертає кількість елементів у контейнері;
   7. boolean contains(String string) повертає true, якщо контейнер містить вказаний елемент;
   8. boolean containsAll(Container container) повертає true, якщо контейнер містить всі елементи з зазначеного у параметрах;
   9. public Iterator<String> iterator() повертає ітератор відповідно до [Interface Iterable](http://docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html).
4. В класі ітератора відповідно до [Interface Iterator](http://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html) реалізувати методи:
   1. public boolean hasNext();
   2. public String next();
   3. public void remove().
5. Продемонструвати роботу ітератора за допомогою циклів while и for each.
6. Забороняється використання контейнерів (колекцій) і алгоритмів з [Java Collections Framework](https://docs.oracle.com/javase/8/docs/technotes/guides/collections/).
   1. **Розробник**: Татаренко Андрій Геннадійович, КІТ-119а, варіант №20.
7. **ОПИС ПРОГРАМИ**
   1. **Засоби ООП**: класи, методи класу.
   2. **Ієрархія та структура класів:** один публічний клас Main, та клас-контейнер з ітератором.
   3. **Важливі фрагменти програми:**

public class Container {

private String[] mas;

private int len;

//---------------------------------------------------

public String toString()

{

StringBuilder str = new StringBuilder();

for(int i = 0; i < len; i++)

{

str.append(mas[i] + " | ");

}

return str.toString();

}

//---------------------------------------------------

public void add(String string)

{

String[] nmas = new String[len + 1];

for(int i = 0; i < len; i++)

{

nmas[i] = mas[i];

}

nmas[len] = string;

len++;

mas = nmas;

}

//---------------------------------------------------

public void clear()

{

for(int i = 0; i < len; i++)

{

mas[i] = null;

}

len = 0;

}

//---------------------------------------------------

public boolean remove(String string)

{

boolean b = false;

for(int i = 0; i < len; i++)

{

if(mas[i].equals(string))

{

b = true;

String[] nmas = new String[len - 1];

for(int j = 0; j < i; j++)

{

nmas[j] = mas[j];

}

for(int j = i; j+1 < len; j++)

{

nmas[j] = mas[j+1];

}

mas = nmas;

len--;

break;

}

}

return b;

}

//---------------------------------------------------

public Object[] toArray()

{

Object[] obj = new Object[len];

for(int i = 0; i < len; i++)

{

obj[i] = mas[i];

}

return obj;

}

//---------------------------------------------------

public int size()

{

return len;

}

//---------------------------------------------------

public boolean contains(String string)

{

boolean con = false;

for(int i = 0; i < len; i++)

{

if(mas[i].equals(string))

{

con = true;

break;

}

}

return con;

}

//---------------------------------------------------

public boolean containsAll(Container container)

{

boolean con = false;

int size = container.size();

int count = 0;

for(int i = 0; i < len; i++)

{

for(int j = 0; j < size; j++)

{

if(mas[i].equals(container.GetEl(j)))

{

count++;

break;

}

}

}

if(count == len)

{

con = true;

}

return con;

}

//---------------------------------------------------

private String GetEl(int i)

{

return mas[i];

}

//---------------------------------------------------

public Container(String... str)

{

len = str.length;

if(len > 0)

{

mas = new String[len];

for(int i = 0; i < len; i++)

{

mas[i] = str[i];

}

}

}

//---------------------------------------------------

public Iterator<String> iterator()

{

return new MyIterator<String>();

}

//---------------------------------------------------

public class MyIterator<String> implements Iterator{

private int ind = 0;

*@Override* public boolean hasNext()

{

if(ind < len) return true;

else return false;

}

*@Override* public Object next()

{

if(ind == len)

{

throw new NoSuchElementException();

}

return mas[ind++];

}

*@Override* public void remove()

{

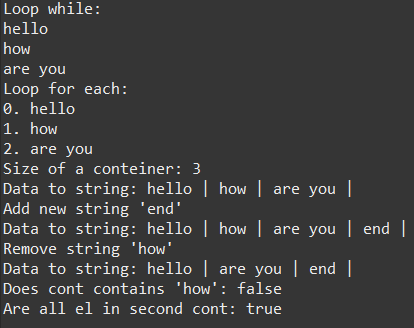
Container.this.remove(mas[--ind]);

}

}

}

**Результат виконання програми**



**Висновки**

# При виконанні даної лабораторної роботи було набуто навички розробки власних контейнерів. Також розробили та застосували власний ітератор.

Програма протестована, виконується без помилок.