**Розробка власних контейнерів. Ітератори**

**Мета роботи**: 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/" \l "_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/).

**Розробник**: Рябов Олексій Володимирович КІТ119а №18.

**Опис програми**

**Засоби ООП**: клас, метод классу.

**Структура класів:** один публічний клас Main, один клас колекція Container

**Важливі фрагменти програми:**

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

Container container = **new** Container("Hi user.","This lab aims to show how I can deal with the container development problem.","All this is needed so that I can keep the lines intact.");

Iterator<String> it = container.getIterator() ;

**for** (;it.hasNext();)

System.***out***.println(it.next());

it = container.getIterator();

**while** (it.hasNext())

System.***out***.println(it.next());

System.***out***.println("Removing the first similar item from the container and displaying it using the method toString :");

System.***out***.println("Result checking - " + container.remove("Hi user."));

System.***out***.println(container.toString());

System.***out***.println("Size of the container - " + container.size());

System.***out***.println("Contains test with string: " + "All this is needed so that I can keep strings with palindromes safe and sound - " + container.contains("All this is needed so that I can keep strings with palindromes safe and sound."));

System.***out***.println("Add one string in my conteiner");

container.add("adda wad ss.");

System.***out***.println("Show : " + container.toString());

System.***out***.println("\nContains all text - " + container.containsAll(**new** Container("adda wad ss.", "This lab aims to show how I can deal with the container development problem.","All this is needed so that I can keep strings with palindromes safe and sound.")));

System.***out***.println("Clearing the container");

container.clear();

}

**import** java.util.Iterator;

**public** **class** Container {

**private** String [] container;

**private** **int** size;

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

{

String str = "";

**for** (String string : container) {

str += string + " ";

}

**return** str;

}

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

{

**int** size = container.length;

String [] new\_container = **new** String[size+1];

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

new\_container[i]=container[i];

}

new\_container[size]=str;

size++;

container = new\_container;

}

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

{

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

container[i]=**null**;

}

size =0;

}

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

{

**boolean** flag = **false**;

String [] new\_container = **new** String[size-1];

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

**if**(container[i].equals(str))

flag = **true**;

}

**if**(flag) {

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

**if**(container[i].equals(str))

i++;

new\_container[j]=container[i];

j++;

}

size--;

container = new\_container;

**return** flag;

}

**else**

{

**return** flag;

}

}

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

{

**return** container;

}

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

{

**return** size;

}

**public** **boolean** containsAll(Container cont) //повертає true, якщо контейнер містить всі елементи з зазначеного у параметрах;

{

**int** count = 0;

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

**for** (**int** j = 0; j < cont.container.length; j++) {

**if**(cont.container[j].equals(container[i]))

count++;

}

}

**if**(count == cont.container.length)

**return** **true**;

**else**

**return** **false**;

}

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

{

**boolean** flag = **false**;

**for** (String string : container) {

**if**(string.equals(str))

flag=**true**;

}

**return** flag;

}

**public** Container(String... str) {

**if**(str.length!=0) {

size = str.length;

container = **new** String[size];

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

container[i]=str[i];

}

}

}

**public** Iterator<String> getIterator() {

**return** **new** My\_iterator<String>();

}

**public** **class** My\_iterator<String> **implements** Iterator {

**int** index;

@Override

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

**return** index < size ? **true** : **false**;

}

@Override

**public** Object next() {

**return** container[index++];

}

/\*Method that removes from the underlying collection the last element returned by this iterator\*/

@Override

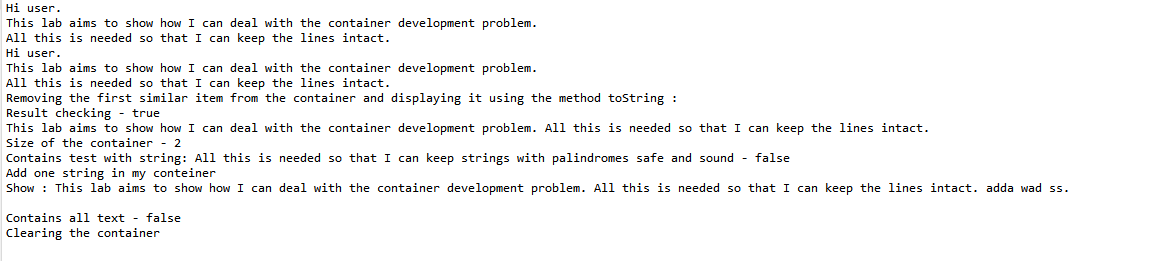
**public** **void** remove() {

Container.**this**.remove(container[--index]);

}

}

**Результати роботи**

****

**Висновки**

Оволодів навичками розробки власної колекції та ітератора.