**Звіт**

**Лабораторна работа 5. Розробка власних контейнерів. Ітератори**

**Мета роботи**:

* Набуття навичок розробки власних контейнерів.
* Використання ітераторів.

**ВИМОГИ**

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.

1. В класі ітератора відповідно до Interface Iterator 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.
   1. **Розробник**: Капелька Ярослав Іванович, КІТ119-а, варіант №9.
3. **ОПИС ПРОГРАМИ**
   1. **Засоби ООП**: клас, метод класу, поле класу.
   2. **Ієрархія та структура класів:** один публічний клас Main та публічний клас MyContainer, у полі якого знаходиться приватний класс MyIterator.
   3. **Важливі фрагменти програми:**

**public** **class** MyContainer {

**private** String[] array;

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

**public** String toString()

{

StringBuilder string = **new** StringBuilder();

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

string.append(array[i] + " ");

**return** string.toString();

}

**public** **void** add(String string)

{

String[] newArray = **new** String[size+1];

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

newArray[i] = array[i];

size++;

newArray[size - 1] = string;

array = newArray;

}

**public** **void** clear()

{

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

array[i] = **null**;

size = 0;

}

**public** **boolean** remove(String string)

{

**boolean** result = **false**;

**int** position = 0;

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

**if**(array[i].equals(string))

{

result = **true**;

position = i;

**break**;

}

**if**(result)

{

String[] newArray = **new** String[size-1];

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

newArray[i] = array[i];

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

newArray[i]=array[i+1];

size--;

array=newArray;

}

**return** result;

}

**public** Object[] toArray()

{

Object[] object = **new** Object[size];

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

object[i]=array[i];

**return** object;

}

**public** **int** size()

{

**return** size;

}

**public** **boolean** contains(String string)

{

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

**if** (array[i].equals(string))

**return** **true**;

**return** **false**;

}

**public** String getLine(**int** index)

{

**return** array[index];

}

**public** **boolean** containsAll(MyContainer container)

{

**boolean** result = **false**;

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

{

result = **false**;

**for** (**int** j = 0; j < container.size(); j++)

{

**if**(array[i].equals(container.getLine(j)))

{

result = **true**;

**break**;

}

}

**if**(!result)

{

**return** **false**;

}

}

**return** result;

}

**public** MyContainer(String... strings)

{

**if**(strings.length > 0)

{

size = strings.length;

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

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

array[i]=strings[i];

}

}

**public** MyIterator<String> getIterator()

{

**return** **new** MyIterator<String>();

}

**private** **class** MyIterator<String> **implements** Iterator {

**int** index;

@Override **public** **boolean** hasNext()

{

**if**(index < size)

**return** **true**;

**else**

**return** **false**;

}

@Override **public** Object next()

{

**if**(index==size)

**throw** **new** NoSuchElementException();

**return** array[index++];

}

@Override **public** **void** remove()

{

MyContainer.**this**.remove(array[--index]);

}

}

}

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

MyContainer array = **new** MyContainer("Привет", "Меня зовут Ярослав", "Мне 18 лет.");

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

String line;

System.***out***.println("Output array using while():");

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

{

line = it.next();

System.***out***.println(line);

}

System.***out***.println("\nДобавление нового элемента");

array.add("Я люблю проводить время на свежем воздухе.");

System.***out***.println("\nOutput array using for:");

it = array.getIterator();

**for**(**int** i = 1;it.hasNext();i++)

{

line = it.next();

System.***out***.println(i + ". " + line);

}

System.***out***.print("\nНахождение строки \"Меня зовут Ярослав\": " + array.contains("Меня зовут Ярослав"));

System.***out***.println("\n\n\r\n" + "Массив содержит все строки данных из другого массива: " + array.containsAll(**new** MyContainer("Привет", "Я люблю проводить время на свежем воздухе.", "Меня зовут Ярослав", "Мне 18 лет.", "Сегрдня отличная погода и хорошее настроение")));

System.***out***.println("\nРазмер массива: " + array.size() + "\nУдаление одного элемента");

array.remove("Я люблю проводить время на свежем воздухе.");

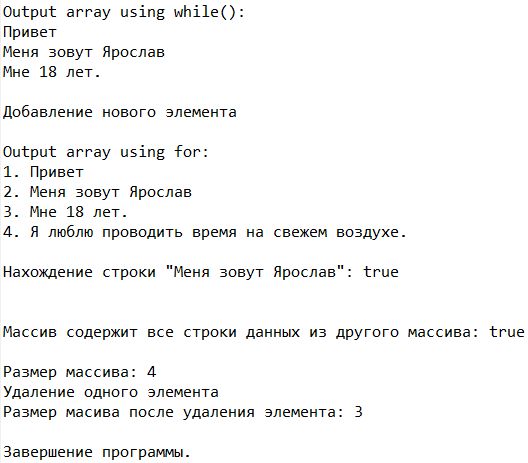
System.***out***.println("Размер масива после удаления элемента: " + array.size() + "\n");

System.***out***.println("Завершение программы.");

array.clear();

}

**Результат виконання програми:**

****

**Висновки**

При виконанні даної лабораторної роботи було набуто практичного досвіду розробки власного контейнера та ітератора.

Програма протестована, виконується без помилок.