**Лабораторна робота №5**

**Розробка власних контейнерів. Ітератори**

**Мета:** Набуття навичок розробки власних контейнерів. Використання ітераторів.

**1 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 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. **Розробник**

* П.І.Б: Заночкин. Є. Д.
* Группа: КІТ-119а
* Варіант: 7

1. **ОПИС ПРОГРАМИ**
   1. **Було використано наступні засоби:**

StringBuilder sb = new StringBuilder() – створення рядку типу StringBuilder;

string.length() – визначення довжини рядка;

Iterator<String> iterator – ітератор.

* 1. **Ієрархія та структура класів**

Було створено 3 класи:

* public class Main – містить метод main;
* public class MyContainer – клас, що містить методи для роботи з контейнером;
* public class MyIterator – клас, що містить методи для роботи з ітератором.
  1. **Важливі фрагменти програми**

public class MyContainer

{

private String[] arrayStr;

private int size;

/\*

\* Конструктор класу MyContainer

\*/

public MyContainer(String... str)

{

if (str.length != 0)

{

size = str.length;

arrayStr = new String[size];

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

arrayStr[i] = str[i];

}

}

/\*

\* Метод toString, який повертає вміст контейнера у вигляді рядка

\*/

public String toString()

{

StringBuilder sb = new StringBuilder();

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

sb.append(arrayStr[i] + " ");

return sb.toString();

}

/\*

\* Метод add, який додає рядок в кінець масиву

\*/

public void add(String string)

{

String newArr[] = new String[size + 1];

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

newArr[i] = arrayStr[i];

newArr[size] = string;

size++;

arrayStr = newArr;

}

/\*

\* Метод clear, який очищує контейнер

\*/

public void clear()

{

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

arrayStr[i] = null;

size = 0;

}

/\*

\* Метод remove, який видаляє перший випадок вказаного елемента з контейнера

\*/

boolean remove(String string)

{

boolean flag = false;

int pos = 0;

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

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

{

flag = true;

pos = i;

break;

}

if (flag)

{

String newArr[] = new String[size - 1];

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

newArr[i] = arrayStr[i];

for (int i = pos, j = pos + 1; j < size; i++, j++)

newArr[i] = arrayStr[j];

size--;

arrayStr = newArr;

}

return flag;

}

/\*

\* Метод toArray, який повертає масив, що містить всі елементи у контейнері

\*/

public Object[] toArray()

{

Object[] arr = new Object[size];

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

arr[i] = arrayStr[i];

return arr;

}

/\*

\* Метод size, який повертає кількість елементів у контейнері

\*/

public int size()

{

return size;

}

/\*

\* Метод contains, який повертає true, якщо контейнер містить вказаний елемент

\*/

public boolean contains(String string)

{

for (String str : arrayStr)

if (str.equals(string))

return true;

return false;

}

/\*

\* Метод containsAll, який повертає true, якщо контейнер містить всі елементи з зазначеного у параметрах

\*/

public boolean containsAll(MyContainer container)

{

if (size < container.size() || container.size() == 0 || size == 0)

return false;

int count = 0;

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

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

if(arrayStr[i].equals(container.arrayStr[j]))

{

count++;

break;

}

if(count == container.size())

return true;

else

return false;

}

/\*

\* Метод iterator, який повертає ітератор відповідно до Interface Iterable.

\*/

public Iterator<String> iterator()

{

return new MyIterator<String>();

}

public class MyIterator<String> implements Iterator

{

int index;

/\*

\* Method that returns true if the iteration has more elements

\*/

@Override

public boolean hasNext()

{

return index < size;

}

/\*

\* Method that returns the next element in the iteration

\*/

@Override

public Object next()

{

return arrayStr[index++];

}

/\*

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

\*/

@Override

public void remove()

{

MyContainer.this.remove(arrayStr[index - 1]);

}

}

}

1. **Результати роботи програми**

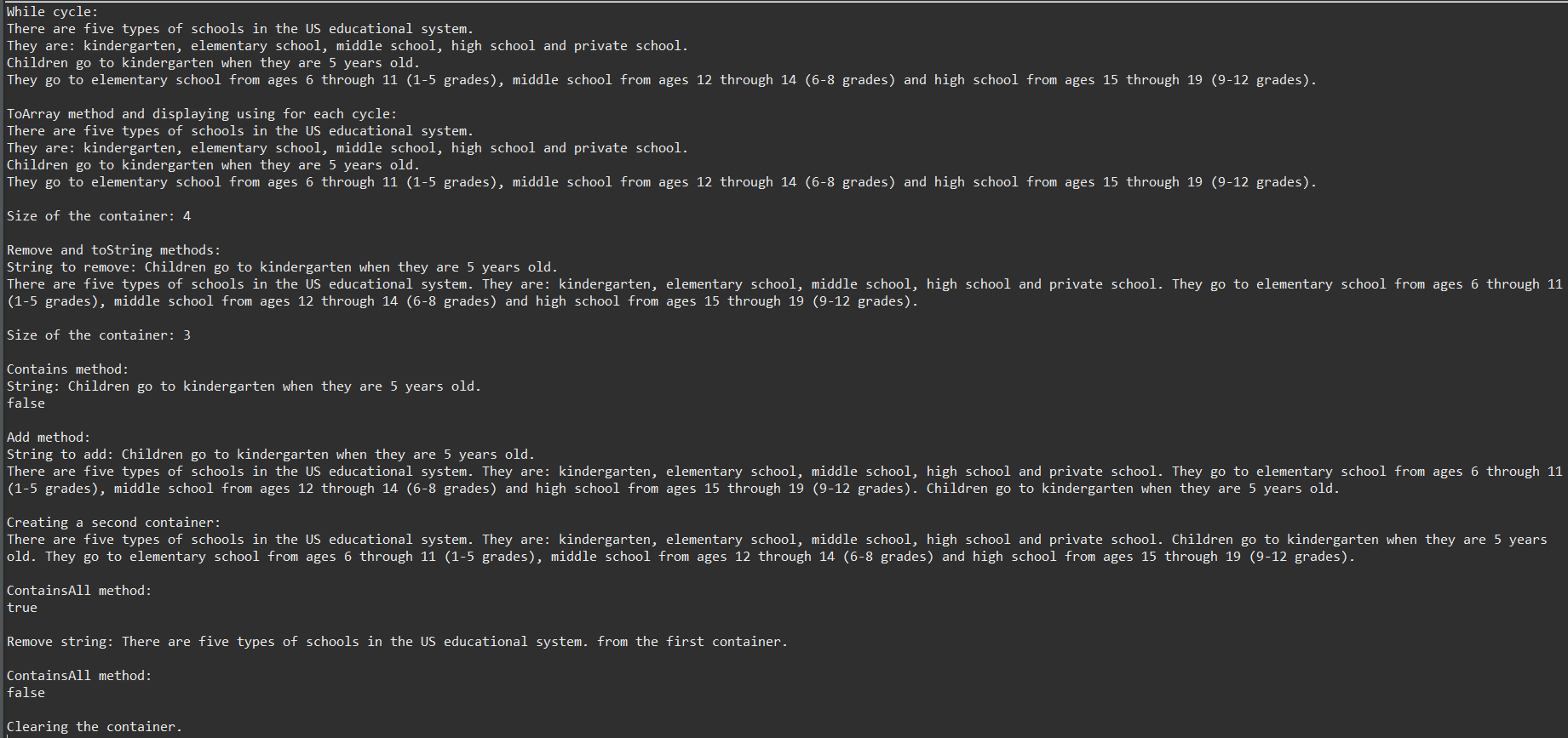
****

Рисунок 5.1 – Результат роботи програми у середовищі Eclipse

**Висновок**

Під час виконання лабораторної роботи було набуто навичок розробки власних контейнерів та роботи з ітераторами у середовищі Eclipse IDE.