Лабораторна робота №5. Розробка власних контейнерів. Ітератори

Мета:

* + Набуття навичок розробки власних контейнерів.
  + Використання ітераторів.

1 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.
3. В класі ітератора відповідно до Interface Iterator 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.

1.1 Розробник

* Абдулаєв Ібрагім Заурбекович
* Група КІТ-119В
* Варіант 1

2 ОПИС ПРОГРАМИ

2.1 Опис класів

* **ua.khpi.oop.abdulaev05.Container**

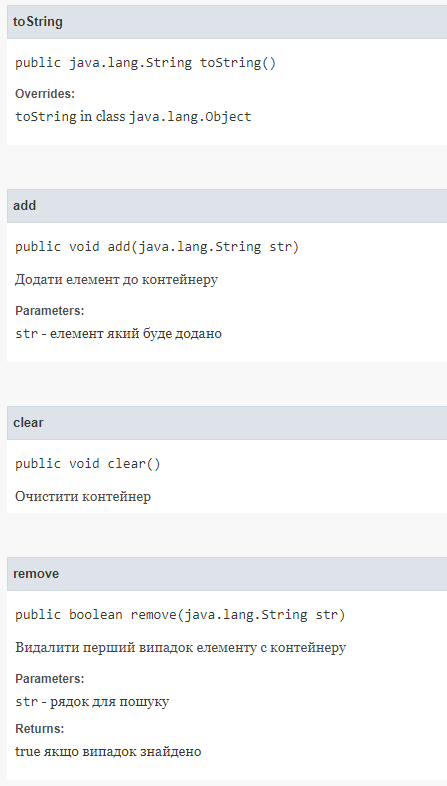


Рис. 2.1 – Опис класу **Container**

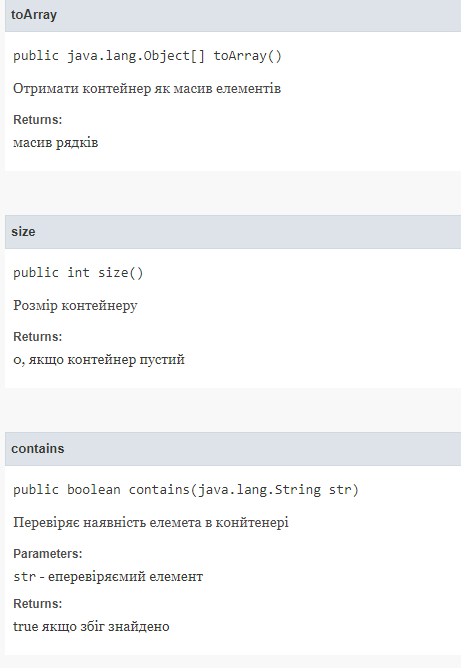


Рис. 2.2 – Опис класу **Container**

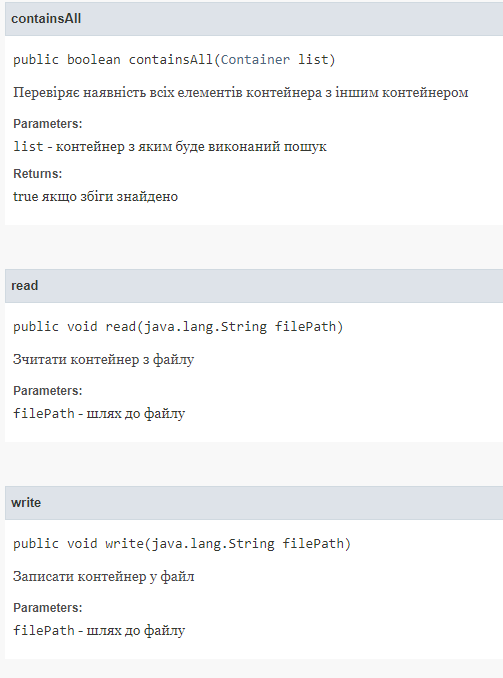


Рис. 2.3 – Опис класу **Container**

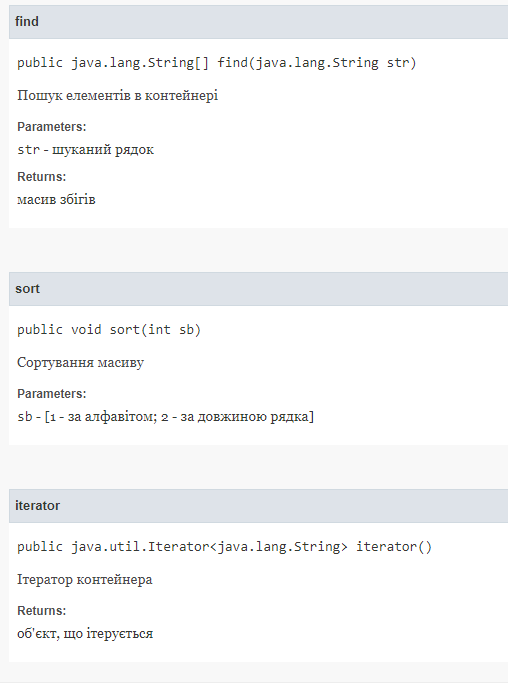


Рис. 2.4 – Опис класу **Container**

* **ua.khpi.oop.abdulaev05.Main**

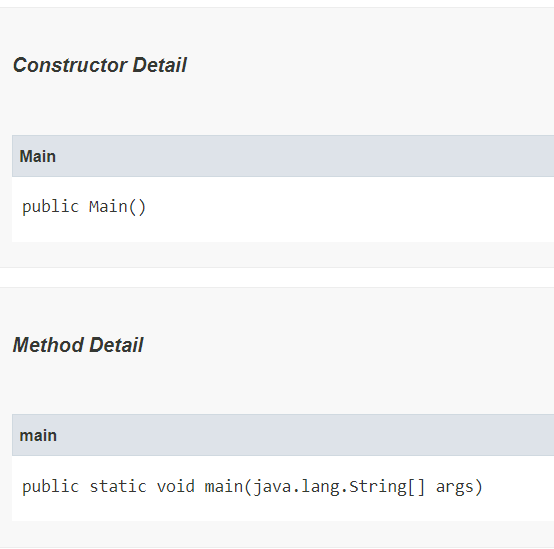


Рис. 2.5 – Опис класу **Main**

2.2 Текст програми

**Main.java**

package ua.khpi.oop.abdulaev05;  
  
import java.util.Iterator;  
  
public class Main {  
  
 public static void main(String[] args) {  
 Container container = new Container();  
  
 container.add("Stroka 1");  
 container.add("Stroka 2");  
 container.add("Stroka 3");  
  
 for (Iterator<String> it = container.iterator(); it.hasNext();) {  
 String str = it.next();  
 System.*out*.println(str);  
 }  
  
 container.remove("Stroka 2");  
  
 System.*out*.println();  
  
 for (Iterator<String> it = container.iterator(); it.hasNext();) {  
 String str = it.next();  
 System.*out*.println(str);  
 }  
  
 container.add("Stroka 4");  
  
 System.*out*.println();  
  
 for (Iterator<String> it = container.iterator(); it.hasNext();) {  
 String str = it.next();  
 System.out.println(str);  
 }  
  
 System.out.printf("\nContainer contains Stroka 4: %b\n", container.contains("Stroka 4"));  
 }  
}

**Container.java**

package ua.khpi.oop.abdulaev05;  
import java.io.\*;  
import java.util.Iterator;  
*/\*\*  
 \* Клас контейнер  
 \*/*public class Container {  
  
 private String[] arr;  
  
 public Container() {  
 arr = new String[]{};  
 }  
  
 public String toString() {  
 StringBuilder builder = new StringBuilder();  
 for (String s : arr)  
 builder.append(s).append(" ");  
 return builder.toString();  
 }  
  
 public void add(String str) {  
 int i;  
 String[] oldArr = arr;  
 arr = new String[oldArr.length + 1];  
 for (i = 0; i < oldArr.length; i++)  
 arr[i] = oldArr[i];  
 arr[i] = str;  
 }  
  
 public void clear() {  
 arr = new String[]{};  
 }  
  
 public boolean remove(String str) {  
 String[] oldArr = arr;  
 if (!contains(str)) return false;  
 arr = new String[oldArr.length - 1];  
 for (int i = 0, j = 0; i < oldArr.length; i++)  
 if (!oldArr[i].contains(str))  
 arr[j++] = oldArr[i];  
 return true;  
 }  
  
 public Object[] toArray() {  
 return arr;  
 }  
  
 public int size() {  
 return arr.length;  
 }  
  
 public boolean contains(String str) {  
 for (String s : arr)  
 if (s.equals(str)) return true;  
  
 return false;  
 }  
  
 public boolean containsAll(Container list) {  
 int i, j;  
 for (i = 0; i < list.size(); i++) {  
 for (j = 0; j < arr.length; j++)  
 if (list.arr[i].equals(arr[j])) break;  
 if (j == arr.length) return false;  
 }  
 return true;  
 }  
  
 public void read(String filePath) {  
 try {  
 FileInputStream fileInputStream = new FileInputStream(filePath.replace("file:", ""));  
 ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);  
  
 arr = (String[]) objectInputStream.readObject();  
 } catch (IOException | ClassNotFoundException e) {  
 System.*out*.println("File not found " + filePath);  
 }  
 }  
  
  
 public void write(String filePath) {  
 try {  
 FileOutputStream outputStream = new FileOutputStream(filePath.replace("file:", ""));  
 ObjectOutputStream objectOutputStream = new ObjectOutputStream(outputStream);  
  
 objectOutputStream.writeObject(arr);  
 objectOutputStream.close();  
 } catch (IOException e) {  
 System.*out*.println("File not found " + filePath);  
 }  
 }  
  
 public String[] find(String str) {  
 String[] results = new String[]{};  
  
 if (str == null) return results;  
  
 for (String s : arr) {  
 if (s.contains(str)) {  
 String[] \_re = results;  
 results = new String[\_re.length + 1];  
 int i;  
  
 for (i = 0; i < \_re.length; i++)  
 results[i] = \_re[i];  
  
 results[i] = s;  
 }  
 }  
  
 return results;  
 }  
  
 public void sort(int sb) {  
 int i, j;  
 String tmp;  
  
 switch (sb) {  
 case 1:  
 for (i = 0; i < arr.length; i++) {  
 for (j = i + 1; j < arr.length; j++) {  
 if (arr[i].length() > arr[j].length()) {  
 tmp = arr[i];  
 arr[i] = arr[j];  
 arr[j] = tmp;  
 }  
 }  
 }  
 break;  
 case 2:  
 for (i = 0; i < arr.length; i++) {  
 for (j = i + 1; j < arr.length; j++) {  
 if (arr[i].compareTo(arr[j]) > 0) {  
 tmp = arr[i];  
 arr[i] = arr[j];  
 arr[j] = tmp;  
 }  
 }  
 }  
 break;  
 }  
 }  
  
 public Iterator<String> iterator() {  
 return new Iterator<String>() {  
 private int index = 0;  
  
 @Override  
 public boolean hasNext() {  
 return index < size() && arr[index] != null;  
 }  
  
 @Override  
 public String next() {  
 return arr[index++];  
 }  
  
 @Override  
 public void remove() {  
 Container.this.remove(arr[index]);  
 }  
 };  
 }  
}

3 ВИСНОВКИ

На лабораторній роботі навчилися розробляти власні контейнери, набули

навичок з використання ітераторів.

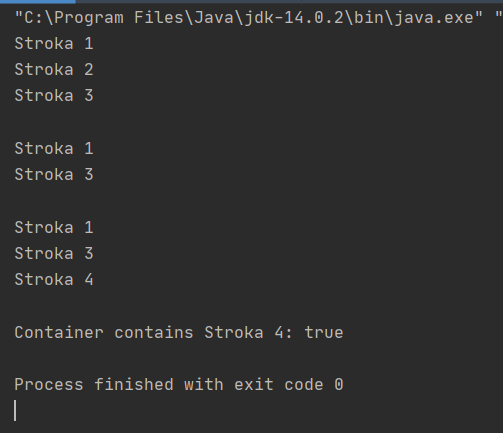


Рис. 3.1 – Результат роботи програми