Лабораторна робота №5

Розробкавласнихконтейнерів. 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:
    - StringtoString() повертає вміст контейнера у вигляді рядка;
    - voidadd(Stringstring) додає вказаний елемент до кінця контейнеру;
    - voidclear() видаляє всі елементи з контейнеру;
    - booleanremove(Stringstring) видаляє перший випадок вказаного елемента з контейнера;
    - Object[] toArray() повертає масив, що містить всі елементи у контейнері;
    - intsize() повертає кількість елементів у контейнері;
    - booleancontains(Stringstring) повертає true, якщо контейнер містить вказаний елемент;
    - booleancontainsAll(Containercontainer) повертає true, якщо контейнер містить всі елементи з зазначеного у параметрах;
    - publicIterator<String>iterator() повертає ітератор відповідно до InterfaceIterable.
3. В класі ітератора відповідно до InterfaceIterator реалізувати методи:
    - publicbooleanhasNext();
    - publicStringnext();
    - publicvoidremove().

4. Продемонструвати роботу ітератора за допомогою циклів while и foreach.
5. Забороняється використання контейнерів (колекцій) і алгоритмів з JavaCollections Framework.

1.1Розробник

- П.І.Б:Наймитенко С.І.

- Група: КІТ-119а

- Варіант 15

2 ОПИС ПРОГРАМИ

2.1Було використано наступні засоби:

- StringBuildersb = newStringBuilder()–створення StringBuilder;
- String.length() – Визначення довжини змінної типу StringBuffer;
- Iterator<String> iterator = MyContainer.getIterator() – Ітератор;

2.2 Ієрархія та структура класів

Було створено 3 класи:

- publicclassMyContainer – клас, що реалізує методи контейнеру.
- publicclassMyIterator – клас, що реалізує методи ітератора.
- public class Main – містить лише метод main.

Важливі фрагменти програми

Клас Container

```
package ua.khpi.oop.naimytenko05;

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]);
		}
	}
}
```

Результат роботи програми:

```
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 strings with palindromes safe and sound.
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 strings with palindromes safe and sound.
Removing the first similar item from the container and displaying it using the method toString :
Result checking - true
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.
Size of the container - 2
Contains test with string: All this is needed so that I can keep strings with palindromes safe and sound - true
Add one string in my container
Show : 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. adda wad ss.

Contains all text - true
Clearing the container
```

ВИСНОВКИ

У результаті виконання лабораторної роботи було набуто навичок з розробки власних контейнерів, роботи з ітераторами у середовищі JavaEclipse.