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

- Дем’яненко Дмитро Андрійович
- Группа: КІТ-119д
- Варіант: 7

2 ОПИС ПРОГРАМИ

2.1 Було використано наступні засоби:

Iterator<String> iterator – ітератор.

2.2 Ієрархія та структура класів

Було створено 3 класи:

- public class Main – містить метод main;
- public class Container – клас, що містить методи для роботи з контейнером;
- public inteface MyIterator – клас, що містить методи для роботи з ітератором.

2.3 Важливі фрагменти програми

```
package ua.khpi.oop.demianenko05;

import java.util.Objects;
public class Container<T> {
  private Node<T> firstNode;
    private Node<T> lastNode;
    private int size = 0;

    public Container() {
        lastNode = new Node<T>(null, firstNode, null);
        firstNode = new Node<T>(null, null, lastNode);
    }

    private class Node<T> {

        private T thisElement;
        private Node<T> nextElement;
        private Node<T> backElement;


        public Node(T thisElement, Node<T> backElement, Node<T> nextElement){
            this.thisElement = thisElement;
            this.nextElement = nextElement;
            this.backElement = backElement;
        }

        public T getThisElement() {
            return thisElement;
        }

        public void setThisElement(T thisElement) {
            this.thisElement = thisElement;
        }

        public Node<T> getNextElement() {
            return nextElement;
        }

        public void setNextElement(Node<T> nextElement) {
            this.nextElement = nextElement;
        }
```

```
        public Node<T> getBackElement() {
            return backElement;
        }

        public void setBackElement(Node<T> backElement) {
            this.backElement = backElement;
        }
    }

    private Node<T> getNode(int index) {
        Node<T> element;//System.out.println(element.getThisElement());

        if(index > size || index < 0){
            System.exit(3);
        }

        if(index <= size/2) {
            element = firstNode.getNextElement();
            for (int i = 0; i < index; i++) {
                element = getNext(element);
            }
        } else {
            element = lastNode.getBackElement();
            for (int i = size-1; index < i; i--) {
                element = getBack(element);
            }
        }

        return element;
    }

    private Node<T> getNext(Node<T> current){
        return current.getNextElement();
    }

    private Node<T> getBack(Node<T> current){
        return current.getBackElement();
    }

    public void add(T newElement){
        Node<T> back = lastNode;
        back.setThisElement(newElement);//System.out.println(back.getThisElement());
        lastNode = new Node<T>(null, back, null);
        back.setNextElement(lastNode);
        size++;
    }

    public T get(int index){
        Node<T> element = getNode(index);
        return element.getThisElement();
    }

    public int getSize(){
        return size;
    }

    public boolean remove(int index){
        Node<T> x = getNode(index);
        Node<T> prevX =
x.getBackElement();//System.out.print(prevX.getThisElement());
        Node<T> nextX =
x.getNextElement();//System.out.print(nextX.getThisElement());
```

```
        if(prevX == null){
            firstNode.setNextElement(nextX);
        } else {
            prevX.setNextElement(nextX);
            x.setNextElement(null);
        }

        if(nextX == null){
            lastNode.setBackElement(prevX);
        } else {
            nextX.setBackElement(prevX);
            x.setBackElement(null);
        }

        size--;

        return true;
    }

    public void clear() {
        Node<T> a = firstNode;
            for (int i = 0; i < size; i++) {
              a.setThisElement(null);
              a.getNextElement();
            }
            size = 0;
    }


    public boolean contains(T t){
        Node<T> element = firstNode.getNextElement();
        for (int index = 0; index < size; index++){
            if(Objects.equals(element.getThisElement(), t)){
                return true;
            } else {
                element = getNext(element);
            }
        }
        return false;
    }



    String printContainer(){
        String s = "";
        for(int i = 0; i < size; i++){
            s = s + " " + get(i).toString();
        }
        return s;
    }

    @Override
    public String toString(){
        return printContainer();
    }
    public MyIterator iterator(){
            return new ListIterator();
    }
       public class ListIterator implements MyIterator {
            int index = 0;
```

```
@Override
public boolean hasNext() {
	if (index<size) {
		return true;
	}
	else return false;
}

@Override
public T next() {
	Node<T> element = getNode(index++);
  return element.getThisElement();
}
}
}
```

Результати роботи програми

```
Добавление элементов команда add
Вывод с помощью команды toString
 The weather very good
Удаление элементов с помощью команды remove
true
 The very good
Количество элементов
3
Есть ли элемент в контейнере 1)The 2)weather
true
false
Использование итератора
The
very
good
Удаление всех элементов.
```

Висновок

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