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:

- Кедровський Максим
- КІТ-119а

- 10 варіант

1.2. Загальне завдання:

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. Опис програми:

```java
package task05;

public class Container {
    private String[] text;
    private int size;
    public Container() {
        text = new String[0];
        size = 0;
    };
    public Container(String input) {
        text = new String[1];
        size = 0;
        int start = 0;
        int end = 0;
        for(int i = 0; i < input.length(); i++) {
            char cur = input.charAt(i);
            if(cur == ' ') {
                end = i;
                add(input.substring(start, end));
                start = i+1;
            }
        }
        add(input.substring(start, input.length()));
    };
    public String toString() {
        String result = "";
        for(int i = 0; i < size; i++) {
            result += text[i] + " ";
        }
        return result;
    }
    public void add(String line) {
        if (size < text.length) {
            text[size++] = line;
        }
        else {
            String[] newlist = new String[size+1];
            for(int i = 0;i<size;i++) {
                newlist[i] = text[i];
            }
            newlist[size++] = line;
            text = newlist;
        }
    }
```

```
    public void clear() {
        text = new String[0];
        size = 0;
    }
    public boolean remove(String line) {
        for (int i = 0; i < size; i++) {
            if(text[i].equals(line)) {
                String[] newlist = new String[size-1];
                for(int a = 0,j = 0;a<size;a++) {
                    if (a != i) {
                        newlist[j++] = text[a];
                    }
                }
                text = newlist;
                size--;
                return true;
            }
        }
        return false;
    }
    public String[] toArray() {
        return text;
    }
    public int size() {
        return size;
    }
    public Iterator iterator(){
        return new ListIterator();
    }
    public class ListIterator implements Iterator {
        int index = 0;

        @Override
        public boolean hasNext() {
            if (index<text.length) {
                return true;
            }
            else return false;
        }

        @Override
        public String next() {
            return text[index++];
        }
```

```
        @Override
        public void remove() {
            String[] newlist = new String[size-1];
            for(int i = 0,j = 0;i<size;i++) {
                if (i != index) {
                    newlist[j++] = text[i];
                }
            }
            text = newlist;
        }
    }
    public boolean contains(String string) {
        for(int i = 0; i < size; i++) {
            if(string.equalsIgnoreCase(text[i])) {
                return true;
            }
        }
        return false;
    }
    public boolean containsAll(Container container) {
        Iterator itr = container.iterator();
        while (itr.hasNext()) {
            if(!contains(itr.next())) return false;
        }
        return true;
    }
}
```

Рисунок 3.1 - Код Container.java

```java
package task05;

public class Main {
    public static void main(String[] args) {
        Container test = new Container("Hello world text yay let's try");
        test.add("OI");
        System.out.println(test.toString());
        test.remove("text");
        System.out.println(test.toString());
        Iterator itr = test.iterator();
        System.out.println(itr.next());
        System.out.println(itr.next());
        System.out.println(itr.next());
        System.out.println(test.contains("world"));
        System.out.println(test.contains("nope"));
        System.out.println(test.containsAll(new Container("Hello yay try")));
    }
}

```

Рисунок 3.2 - Код Main.java

Було розроблено клас Container, що містить методи роботи з змістом, ітератором. Також розроблено власний класс Iterator, подібний до Interface Iterable.

3. Варіанти використання:

Реалізований функціонал дозволяє працювати з контейнерами текстових строк, ітераторами. У Main демонструються методи Container.

```
<terminated> Main (4) [Java Application] C:\J
Hello world text yay let's try OI
Hello world yay let's try OI
Hello
world
yay
true
false
true
```

Рисунок 3.3 - Результат роботи Main.class

Висновок:

Набув навички розробки власних контейнерів та використання ітераторів. Мовою Java було розроблено програму відповідно до індивідуального завдання.