Project task-jtr-practice2

```
Практическое задание №2
_____________________


Задание 1

1.1. Создать интерфейс MyList следующего содержания:

-----------------------------------------------------
public interface MyList {
    // appends the specified element
    // to the end of this list
    void add(Object e);

    // removes all of the elements from this list
    void clear();

    // removes the first occurrence of the specified
    // element from this list
    boolean remove(Object o);

    // returns an array containing all of the elements
    // in this list in proper sequence
    Object[] toArray();

    // returns the number of elements in this list
    int size();

    // returns true if this list contains
    // the specified element
    boolean contains(Object o);

    // returns true if this list contains all
    // of the elements of the specified list
    boolean containsAll(MyList c);
}
-----------------------------------------------------

1.2. Создать класс MyListImpl, который реализует MyList.

1.3. Переопределить метод toString таким образом, чтобы результат выводился
в виде:

-----------------------------------------------------
[e.toString(), e2.toString(), ... ]
-----------------------------------------------------

где e, e2, ... - элементы контейнера

1.4. Создать класс Demo, который демонстрирует работу всех методов
контейнера (см. замечания внизу).

_____________________

Задание 2

2.1. Добавить к интерфейсу MyList наследование интерфейса Iterable<Object>
(java.lang.Iterable).

-----------------------------------------------------
MyList extends Iterable<Object>
```

```
------------------------------------------------------

Реализовать в контейнере MyListImpl метод
------------------------------------------------------
public Iterator<Object> iterator() {
    return new IteratorImpl();
}
------------------------------------------------------

который возвращает объект внутреннего класса IteratorImpl:

------------------------------------------------------
private class IteratorImpl implements Iterator<Object> {

    // returns true if the iteration has more elements
    public boolean hasNext() {
        // ...
    }

    // returns the next element in the iteration
    public Object next() {
        // ...
    }

    // removes from the underlying collection
    // the last element returned by this iterator
    public void remove() {
        // ...
    }
}
------------------------------------------------------

Алгоритм в методе remove может быть следующим:

------------------------------------------------------
ЕСЛИ ПЕРЕД ВЫЗОВОМ remove НЕ БЫЛ ВЫЗВАН МЕТОД next
ИЛИ ПЕРЕД ВЫЗОВОМ remove БЫЛ ВЫЗВАН remove (повторный вызов remove)
    ТО ВЫБРОСИТЬ ИСКЛЮЧЕНИЕ (так и вставить в код): throw new
IllegalStateException();
В ДАННОМ МЕСТЕ ОПРЕДЕЛИТЬ И УДАЛИТЬ СООТВЕТСТВУЮЩИЙ ЭЛЕМЕНТ
------------------------------------------------------

2.2. Продемонстрировать работу итератора (см. код класса Demo).

_________________________

Задание 3

3.1. Определить интерфейс ListIterator:

------------------------------------------------------
interface ListIterator extends java.util.Iterator<Object> {
    // returns true if this list iterator has more
    // elements when traversing the list
    // in the reverse direction
    boolean hasPrevious();


    // returns the previous element in the list
    // and moves the cursor position backwards
    Object previous();


    // replaces the last element returned by next
    // or previous with the specified element
    void set(Object e);

    // removes from the list the last element
    // that was returned by next or previous
    void remove();
}
```

```
------------------------------------------------------

Методы set/remove могут быть вызваны только после next/previous
IllegalStateException (см. п. 2.1.)

3.2. Создать интерфейс ListIterable:

------------------------------------------------------
interface ListIterable {
    ListIterator listIterator();
}
------------------------------------------------------

3.3. Добавить к классу MyListImpl реализацию интерфейса ListIterable:

------------------------------------------------------
class MyListImpl implements MyList, ListIterable {...}
------------------------------------------------------

3.4. Добавить в класс MyListImpl метод
------------------------------------------------------
public ListIterator listIterator() {
    return new ListIteratorImpl();
}
------------------------------------------------------

который возвращает объект внутреннего класса ListIteratorImpl:

------------------------------------------------------
private class ListIteratorImpl extends IteratorImpl implements ListIterator
{
    // IMPLEMENT ALL METHODS HERE!!!
}
------------------------------------------------------

3.5. Продемонстрировать работу итератора ListIterator (см. код класса
Demo).

_____________________

Замечания

1. Результат должен быть представлен в виде проекта с именем Practice2.

2. Корневой пакет для всех классов и прочих пакетов (если они потребуются):
ua.nure.your_last_name.Practice2, где your_last_name - ваш логин без кода
проекта.

3. Дополнительно в корневой пакет положить класс Demo, который демонстриует
работу всех подзадач.
Содержимое класса Demo должно быть таким каким оно дано в самом конце
документа.
Скопировать и вставить как есть.
В комментариях - информация, какой вывод должен у вас быть.
Перед заливкой в репозиторий убедиться, что приложение генерирует
правильный результат.

4. Проект загрузить в репозиторий, проверить, успешность сборки в Jenkins,
оптимизировать метрики в Sonar.

_____________________

Содержимое класса Demo
------------------------------------------------------
package ua.nure.your_last_name.Practice2;

import java.util.Iterator;

public class Demo {
    public static void main(String[] args) {
        System.out.println("==== Part1");
```

```
MyList list = new MyListImpl();
// [A, A2]
list.add("A");
System.out.println(list);
// []
list.clear();
System.out.println(list);
// [A, A3]
list.add("A");
list.add("A2");
list.add("A3");
list.remove("A2");
System.out.println(list);
// AA3
for (Object el : list.toArray()) {
    System.out.print(el);
}
System.out.println();
// 2
System.out.println(list.size());
// false
System.out.println(list.contains("B"));
// true
System.out.println(list.contains("A3"));
// true
list.add("A2");
MyList anotherList = new MyListImpl();
anotherList.add("A");
anotherList.add("A2");
System.out.println(list.containsAll(anotherList));
// false
anotherList.add("B");
System.out.println(list.containsAll(anotherList));
// true
list.add("B");
System.out.println(list.containsAll(anotherList));

System.out.println("==== Part2");

list = new MyListImpl();
list.add(1);
list.add(2);
list.add(3);
list.add(4);

// 1 2 3 4
Iterator<Object> it = list.iterator();
while (it.hasNext()) {
    System.out.print(it.next() + " ");
}
System.out.println();
// [1, 3, 4]
it = list.iterator();
it.next();
it.next();
it.remove();
System.out.println(list);

// 3
System.out.println(it.next());

// [1, 4]
it.remove();
System.out.println(list);

// class java.lang.IllegalStateException
try {
    it.remove();
} catch (IllegalStateException ex) {
    System.out.println(ex.getClass());
}
```

```
        System.out.println("==== Part3");

        list = new MyListImpl();
        list.add(2);
        list.add(3);
        list.add(4);

        // 1 2 3 4
        ListIterator lit = ((ListIterable)list).listIterator();
        while (lit.hasNext()) {
            System.out.print(lit.next() + " ");
        }
        System.out.println();
        // 4 3 2 1
        while (lit.hasPrevious()) {
            System.out.print(lit.previous() + " ");
        }
        System.out.println();

        list = new MyListImpl();
        lit = ((ListIterable)list).listIterator();
        // false
        System.out.println(lit.hasNext());

        // false
        System.out.println(lit.hasPrevious());
        // Elemenet
        list.add("Element");
        System.out.println(lit.next());

        // false
        System.out.println(lit.hasNext());

        // true
        System.out.println(lit.hasPrevious());
    }
}
-------------------------------------------------------
```


Project disk usage information + trend graph

Disk Usage: Workspace 0, Builds {all=0, locked=0}, Job directory 20624

Recent Changes

Permalinks