P. Thiemann, L. Fennell Sommersemester 2017

Programmieren in Java

`http://proglang.informatik.uni-freiburg.de/teaching/java/2017/`

undo-array

Array mit Undo-Funktionalität

Woche 11 Aufgabe 1/3

Herausgabe: 2017-07-12
Abgabe: 2017-07-31

Achtung: beachten Sie unbedingt die allgemeinen Hinweise zur Abgabe auf der Homepage.

Project `undo-array`
Package `undoarray`
Klassen

`UndoArray<X>`
`public UndoArray(X init, int size, int historySize)`
`public void put(int idx, X elem)`
`public List<X> get()`
`public boolean undo()`

Die Klasse `UndoArray<X>` implementiert ein generisches Array (also eine Liste mit fester Länge) mit „Undo" Funktionalität: Das Array unterstützt das Überschreiben von Elementen und das Auslesen der Elemente. Außerdem können eine bestimmte Anzahl Schreiboperationen durch Undo-Operationen rückgängig gemacht werden.

- Der Konstruktor nimmt drei Argumente: `init` gibt einen Wert an, mit dem die Felder des Arrays initialisiert werden sollen. `size` gibt die Länge des Arrays an und `historySize` die Anzahl der aufeinanderfolgenden Undo-Operationen, die unterstützt werden. Die Argumente `size` und `historySize` müssen ≥ 0 sein, ansonsten wirft der Konstruktor eine `IllegalArgumentException`.

- Die Methode `put` überschreibt den Wert `elem` an Position `idx`. Sie wirft eine `IndexOutOfBoundsExcepti` wenn der Index ungültig ist.

  Die `put` Operation soll auch in einer *History* gespeichert werden, um die Undo-Funktionalität zu ermöglichen. Die History speicher höchstens `historySize` Operationen.

- Die Methode `get` gibt den Inhalt des Arrays als Liste zurück. Damit die Undo-Funktionalität nicht beeinträchtigt wird, soll diese Liste nicht veränderbar sein. Unnötiges Kopieren soll aber auch vermieden werden; daher wirft die zurückgegebene Liste eine

  `UnsupportedOperationException`

falls versucht wird Sie zu ändern (siehe hierzu `Collections.unmodifiableList` in der Java-Api).

- Die Methode `undo` macht die letzte Schreiboperation in der History rückgängig und gibt dann `true` zurück. Das heißt, nach dem Aufruf von `undo` steht im letzten überschriebenen Feld wieder der Wert, der vor dem Überschreiben dort stand.

  Ist die History leer, gibt der `undo` Aufruf `false` zurück und hat ansonsten keinen Effekt.

Wie die History implementiert ist, bleibt Ihnen überlassen... sie soll aber nicht das gesammte Array speichern.

Beispieltests

```
1  package undoarray;
2
3  import org.junit.Test;
4
5  import java.util.Arrays;
6
7  import static org.junit.Assert.*;
8
9  public class ExampleTests {
10
11     @Test
12     public void testPut() {
13         UndoArray<String> arr = new UndoArray<>("", 3, 5);
14         arr.put(0, "Hello");
15         arr.put(1, "World");
16
17         assertEquals(Arrays.asList("Hello", "World", ""), arr.get());
18     }
19
20     @Test
21     public void testUndo() {
22         UndoArray<String> arr = new UndoArray<>("", 3, 2);
23         arr.put(0, "Hello");
24         arr.put(1, "World");
25         arr.put(2, "!");
26
27         assertEquals(Arrays.asList("Hello", "World", "!"), arr.get());
28
29         assertTrue(arr.undo());
30         assertTrue(arr.undo());
31         assertFalse(arr.undo());
32
33         assertEquals(Arrays.asList("Hello", "", ""), arr.get());
34     }
```

```
35
36      @Test
37      public void testUndo2() {
38          UndoArray<String> arr = new UndoArray<>("", 2, 2);
39          arr.put(0, "Hello");
40          arr.put(1, "World!");
41          arr.put(1, "Welt!");
42
43          assertEquals(Arrays.asList("Hello", "Welt!"), arr.get());
44
45          assertTrue(arr.undo());
46
47          assertEquals(Arrays.asList("Hello", "World!"), arr.get());
48      }
49
50      @Test
51      public void testUndo3() {
52          UndoArray<String> arr = new UndoArray<>("", 2, 10);
53          arr.put(0, "Hello");
54          arr.put(1, "World!");
55          arr.put(1, "Welt!");
56
57          assertEquals(Arrays.asList("Hello", "Welt!"), arr.get());
58
59          assertTrue(arr.undo());
60          assertEquals(Arrays.asList("Hello", "World!"), arr.get());
61
62          arr.put(1, "!");
63          assertTrue(arr.undo());
64          assertEquals(Arrays.asList("Hello", "World!"), arr.get());
65
66          assertTrue(arr.undo());
67
68          assertTrue(arr.undo());
69
70          assertEquals(Arrays.asList("", ""), arr.get());
71      }
72
73  }
```