Grundlagen der Programmkonstruktion
Übungsblatt 5 (zu lösen bis 17./18./19. Juni 2013)

1 Iterator für ein Array

1.1 Iterator erstellen ohne remove-Methode (1 Punkt)

Erstellen Sie eine Klasse `MyStringArrayIterator`, der das Interface `Iterator<String>` erweitert. Die Struktur der Klasse ist dabei wie folgt:

```
public class MyStringArrayIterator implements Iterator<String> {

    // TODO Aufgabe 1.1

    public MyStringArrayIterator(String[] array) {
        // TODO Aufgabe 1.1
    }

    public boolean hasNext() {
        // TODO Aufgabe 1.1
    }

    public String next() {
        // TODO Aufgabe 1.1
    }

    public void remove() {
        // TODO Aufgabe 1.2
    }
}
```

Mit dem Iterator sollen Sie die Elemente im Array, das dem Konstruktor als Parameter übergeben wird, durchiterieren, d.h., der erste Aufruf von `next()` liefert `array[0]`, der zweite `array[1]` etc...

Beachten Sie, dass die Methode `next()` eine Exception werfen soll wenn es kein nächstes Element gibt (am besten ist, sie ignorieren jede Form von Fehlerbehandlung, dann wird eine richtige Exception geworfen). Die Rückgabe von `null` ist keine gute Idee.

Das Verwenden von bereits vorhandenen Iteratoren oder Collections, das Kopieren des Arrays in eine andere Datenstruktur oder ähnliches, ist nicht erlaubt!

1.2 `remove()` (1 Punkt)

Implementieren Sie auch die Methode `remove()`. Sie dürfen und sollen dabei annehmen, dass `remove()` nur aufgerufen wird, nachdem `next()` zumindest einmal

aufgerufen wurde. Zusätzlich wird `remove()` maximal einmal pro Aufruf von `next()` aufgerufen. Fehlerbehandlungen, die diese Fälle behandeln, sind nicht gefordert und wahrscheinlich kontraprodiktiv.

2 Iterable-Interface in Liste (3 Punkt)

Gegeben ist die folgende Listenklasse:

```
public class MyList<A> {
    private class Node {
        A value;
        Node next;

        Node(A value, Node next) {
            this.value = value;
            this.next = next;
        }
    }

    private Node head = null;

    public void add(A a) {
        this.head = new Node(a, this.head);
    }
}
```

Implementieren Sie das `Iterable<A>`-Interface.

Erstellen Sie dazu eine innere Klasse (sie können dadurch auf alle Objektvariablen direkt zugreifen; getter sind nicht nötig), die das Iterator-Interface implementiert.

Der Punkteschlüssel dieser Aufgabe ist folgendermaßen:

1. Interfaces + Klassen richtig erstellt: 0.5 Punkte.
2. next-Methode + hasNext-Methode: je 0.75 Punkte.
3. remove-Methode: 1 Punkt.

Für die Methoden im Iterator beachten Sie

Achten Sie in dieser Methode auf die Effizienz ihres Iterators. Implementieren Sie alle Methoden ausschließlich über die Objektvariablen in der Node-Klasse.

Benutzten Sie *keine* `get(int index)` oder `remove(int index)`-Methoden oder ähnliches. Auch das Kopieren der Datenstruktur ist nicht erlaubt. Ein Aufruf auf den Konstruktor des Iterators und die Methoden `hasNext()`, `next()` und `remove()` muss konstante Laufzeit haben.

Beachten Sie desweiteren die Anmerkungen zur ersten Aufgabe bezüglich Fehlerbehandlung.