Aufgabenblatt #6

Wenn Sie es nicht bereits gemacht haben, lesen Sie bitte die relevanten Teile des Skriptums und erarbeiten Sie gemeinsam die bereits durchgegangen Fragen am Ende des Kapitels. Dieses Aufgabenblatt dient zur Übungsvorbereitung zum zweiten Test am 3. Dezember.

Siehe Skriptum bis etwa Seite 267.

Übungseinheit

Aufgabe 1: Frage überlegen

Ihre Aufgabe für die nächste Übungseinheit am 3. bis 5. Dezember ist es, sich zumindest eine Frage zur Programmierung aufzuschreiben. Da die Übungseinheit für die meisten von Ihnen nach dem Test sein wird, muss sich die Frage nicht unbedingt auf das Testgebiet beziehen.

Cycle

`Cycle` ist eine Datenstruktur, die genauso wie eine einfach verkettete Liste funktioniert. Der einzige Unterschied ist, dass sie dadurch abgeschlossen wird, indem sie – statt auf `null` – auf die erste Instanz der Liste zeigt.

Listing 1: Cycle

```
public class Cycle {

    private int elem;
    private Cycle next;

    public Cycle(int value, Cycle cycle) {
        elem = value;
        if (cycle == null) {
            next = this;
        } else {
            next = cycle.next;
            cycle.next = this;
        }
    }
}
```

Aufgabe 2: Ausgeben

Ergänzen Sie die Klasse `Cycle` um die Methode `String toString()`. Geben Sie in der Methode das eigene Element und alle erreichbaren Elemente von `Cycle` aus. Schreiben Sie zusätzlich noch die Methode `int count()`. Beachten Sie, dass Sie erkennen müssen, wann sich der Kreis wieder schließt, damit die Methoden terminieren.

Aufgabe 3: equals

Ergänzen Sie die Klasse `Cycle` um die Methode `boolean equals(Object o)`. Vergleichen Sie dabei ob `Cycle` genau die gleichen Elemente beinhaltet. Achten Sie auch darauf, was in der Java-API über `equals` steht und behandeln Sie die dort angeführten Sonderfälle. Argumentieren Sie, warum das Ersetzbarkeitsprinzip verletzt ist, wenn diese Punkte nicht berücksichtigt werden. Geben Sie auch eine Methode, welche ohne Berücksichtigung der Sonderfälle nicht mehr funktionieren würde, an.

Die Methode darf *keine* NullPointerException oder ClassCastException werfen, unabhängig davon, welcher Parameter übergeben wird.

Aufgabe 4: Elemente zurückgeben

Schreiben Sie die Methode `int getFirst()`, welche `this.elem` zurückgibt. Nun schreiben Sie Methode `int getIndex(int n)`, welches das n-te Element zurückgeben soll. Schließlich soll die Methode `int getLast()` das letzte Element zurückgeben, also genau jenes, bevor sich der Kreis wieder schließt.

Aufgabe 5: fold

Schreiben Sie die Methode `int fold()`, welche die Summe aller Elemente berechnet und diese zurückgibt.

Aufgabe 6: map

Schreiben Sie die Methode `Cycle map()`, welche ein neues Objekt von `Cycle` retourniert. Dieses soll die gleiche Struktur wie der ganze Zyklus von `this` haben, aber die Elemente sollen jeweils die Reste der Division der ursprünglichen Elemente durch 5 sein.

Aufgabe 7: reduce

Schreiben Sie die Methode `Cycle reduce()`, welche einen neuen `Cycle` zurückliefert, der keine negativen Elemente enthält. Alle nicht-negativen Elemente sollen jedoch in derselben Reihenfolge vorkommen.

Aufgabe 8: Testprogramm

Schreiben Sie die statische Methode `main`, welche zumindest zweimal `Cycle` mit jeweils mindestens fünf Elementen instantiiert. Rufen Sie `map`, `reduce` und dann `fold` auf. Vergleichen Sie nun jeweils zwei Instanzen von `Cycle`, wobei ein Vergleich auch `true` zurückliefern soll. Abschließend geben Sie alle Objekte mit Hilfe von `toString()` aus.

Aufgabe 9: Grafisch veranschaulichen

Zeichnen Sie auf, was genau in Ihrem Testprogramm passiert. Überlegen Sie sich im besonderen genau was der Konstruktor macht. Zeichnen Sie auch auf, wie die einzelnen Objekte zusammenhängen.

Vergleiche Abbildung 4.5 im Skriptum auf Seite 252.

Aufgabe 10: Abschätzung

Schätzen Sie die algorithmische Kosten der obigen Funktionen `int getFirst()` und `int fold(Cycle a)` ab. Dabei soll die durchschnittliche und maximale Laufzeit sowie der Speicherverbrauch betrachtet werden. Verwenden Sie dazu die O-Notation.