Prof. Dr. A. Weber, Dr. G. Kniesel
J. Tautges, P. Bihler
oose.cs.uni-bonn.de

Universität Bonn
Institut für Informatik II
Institut für Informatik III
24.06.2008

Sommersemester 2008

Übungen zur Vorlesung
Objektorientierte Softwareentwicklung (BA-INF-024)
Aufgabenblatt 11

Zu bearbeiten bis: 04.07.2008

Aufgabe 1 *(Stacks, Queues, Bäume - 2 Punkte)*

Überlegen Sie sich für jede Aussage ein Argument, mit dem Sie eindeutig bestimmen können, ob die Aussage wahr oder falsch ist. Für jede richtig angekreuzte Antwort werden 0,5 Punkte angerechnet, für jede falsch angekreuzte Antwort 0,5 Punkte abgezogen. Nicht angekreuzte Antworten geben keinen Punktabzug.

wahr	falsch	
		Die Klasse `Stack` aus dem Java-Paket `java.util` ist eine Erweiterung der Klasse `Vector`.
		Die Methode `peek()` der Klasse `Stack` lässt sich durch die Methoden `push()` und `pop()` ausdrücken.
		Die Methoden `element()` und `peek()` der Klasse `Queue` liefern stets dasselbe Ergebnis.
		Bäume sind Spezialfälle von Listen.

Aufgabe 2 *(Speicherbilder und Listen - 6 Punkte)*

Die Klasse `List` sowie die Funktionen `insertFirst()`, `reverseList()` und `reverseListCon()` seien wie in den Vorlesungsfolien implementiert (vgl. auch Aufgabe 3 im Übungsblatt 10). Gegeben sei folgende `main`-Methode:

```
static public void main (String[] args) {
  List<Integer> s1, s2;
  s1 = new List();
  s1.insertFirst(7);
  s1.insertFirst(8);
  s2 = s1.reverseListCon();
  // Stelle 1

  s1.reverseList();
  // Stelle 2

  s1.insertFirst(2);
  s2.insertFirst(3);
  // Stelle 3
  }
}
```

Skizzieren Sie den Zustand des Speichers an den Stellen 1, 2 und 3!

Aufgabe 3 *(Listen - 6 Punkte)*

Erweitern Sie die (generische) Klasse `List` um die folgenden Methoden:

- `public void concat(List<T> a)` – konkateniert Listen *destruktiv*, d.h. hängt an das Ende der `this`-Liste die Liste `a` an,
- `public List<T> append(List<T> a)` – konkateniert Listen konstruktiv, d.h. es wird eine neue Liste zurückgegeben.

Aufgabe 4 *(Bäume - 11 Punkte)*

a) Skizzieren Sie einen Strukturbaum, der den folgenden Ausdruck repräsentiert:

a+b*c+d

b) Erstellen Sie basierend auf den in der Vorlesung vorgestellten Implementierungen der Klassen `Tree` und `Node` eine Klasse `Demo`, die den Strukturbaum aus Aufgabe 3a) aufbaut. Überprüfen Sie die korrekte Struktur des Baums, indem sie sich die Knoten in Inorder-Reihenfolge ausgeben lassen!

c) Erstellen Sie eine Klasse `CountAction`, die das im aktuellen Foliensatz zu findende Interface `NodeActionInterface` implementiert. Sie soll zwei Variablen für die Anzahl der Knoten mit einem bzw. zwei Nachfolgern und die Methoden `int getOneSucc()` und `int getTwoSucc()`, die die Werte dieser Variablen zurückgeben, enthalten. Die Implementierung der `action`-Methode, die im Interface beschrieben wird, soll beim Zählen der Nachfolgerknoten des als Parameter übergebenen Knotens die Werte der Variablen entsprechend erhöhen.

Aufgabe 5 *(Bonusaufgabe - 5* Punkte)*

**Bonusaufgabe*: Durch Lösen dieser Aufgabe können Sie sich zusätzliche Bonuspunkte verdienen!

Implementieren Sie eine Klasse zur Repräsentation allgemeiner Graphen! Zur Implementierung der Kantenrelation existieren verschiedene Möglichkeiten. Diskutieren Sie Vor- und Nachteile der von Ihnen verwendeten Struktur!