Praktische Informatik 2 SoSe 98

Prof. Bernd Krieg-Brückner
Axel Lankenau
Shi Hui
Christoph Lüth

3. Übungsblatt

Ausgabe: 25.05.98

Abgabe: 15.06.98

---

1 *Stapelverarbeitung* *Wichtung: 30%*

Implementieren Sie eine Klasse `BoundedStack`, die einen in der Länge beschränkten Keller (von ganzen Zahlen) modelliert. Der Konstruktor der Klasse sollte als Argument die Größe des Stacks erhalten. Die Methoden sind `push`, `pop` und `top` sowie `empty`. Für Fehlerfälle, wie das Überschreiten der Obergrenze oder ein `pop` bei leerem Keller, sollen Ausnahmen vorgesehen werden.

2 *Arithmetik verkehrt* *Wichtung: 30%*

Unter Benutzung des Kellers von Aufgabe 1 implementieren Sie einen einfachen Taschenrechner, der umgekehrt polnische Notation (UPN, auch Postfix-Notation genannt) unterstützt. Der Taschenrechner soll Zeilen mit einem Ausdruck von der Eingabe lesen, diesen parsieren, ausrechnen und ein Ergebnis ausgeben. So erzeugt die Eingabe `3 4 +` die Ausgabe `7`, die Eingabe `10 5 6 * 15 + -` die Ausgabe `35`, die Eingabe `3 +` eine Fehlermeldung, und `3 4 8 *` die Ausgabe `32` nebst einer Warnung, daß der Keller nicht leer ist.

Tip: Javas Standardklasse `StringTokenizer` ist hier hilfreich.

3 *Kettenlisten* *Wichtung: 20%*

Implementieren Sie eine Klasse `DoubleLinkedList`, die doppelt verkettete Listen (von ganzen Zahlen) realisiert. Die Klasse soll einen Konstruktor haben, der eine leere Liste erzeugt. Intern hat die Klasse einen Zeiger, der auf das aktuelle Listenelement zeigt. Mit der Methode `get` läßt sich dieses zurückgeben, mit den Methoden `next` und `prev` läßt sich der Zeiger vor- und zurückbewegen. Mit der Methoden `insert` wird ein Element *vor* dem aktuellen Element eingefügt (danach ist das eingefügte Element das aktuelle), und mit `delete` wird das aktuelle Element gelöscht. Für Fehlerfälle (welche?) sind Ausnahmen vorzusehen.

Welche weiteren Methoden wären für diese Klasse sinnvoll, und warum?

4 *Ein Umtrunk in fröhlicher Runde* *Wichtung: 20%*

Verändern Sie ihre Implementation der letzten Aufgabe zu *zyklischen Listen*: der Nachfolger des letzten Elementes in der Liste ist wiederum das erste Element in der Liste.

Mithilfe dieser zyklischen Liste implementieren Sie eine Lösung des "Josephus-Problems": eine Gruppe von N Kultmitgliedern des Ordens der Erleuchteten der Letzten Tage hat beschlossen, Massenselbstmord zu begehen, weil das Ende der Welt bevorsteht. Die Ordensmitglieder sitzen in einem Kreis, von einzelnen Kerzen flackernd beleuchtet, und in der Mitte ist eine mit obskuren Runen verzierte Silberschale strychninverfeinerten Heidelbeersaftes, von dem der jeweils M-te Kultist einen Schluck nimmt, bevor er erleuchtet hintenüber kippt, und die restlichen Mitglieder den Kreis schließen. Das Josephus-Problem besteht darin herauszufinden, wer als letzter überbleibt— und ob er es sich dann nicht noch anders überlegt.

Implementieren sie eine Klasse `Josephus` mit der Methode `exit_order`, die zwei Argumente M und N hat, und die Exitusreihenfolge der Kultmitglieder ausgibt. Für $N = 9$, $M = 5$ wäre diese $5, 1, 7, 4, 3, 6, 9, 2, 8$.