Übungen zur Einführung in die Programmiersprache Java

Universität Regensburg
NWF II - Physik
Dominik Köppl

Wintersemester 2011/12
Blatt 11

---

41 Stack

(a) Entwerfen Sie eine Funktion `Stack<T> invertiereStack(Stack<T> stack);`. Diese Methode erhält einen Stack `stack` und liefert einen Neuen zurück, der alle Elemente des ursprünglichen Stacks in umgekehrter Reihenfolge enthält.

(b) Schreiben Sie eine Funktion `Stack<T>[] divideStack(Stack<T> stack, T delimiter);`. Diese Methode soll den Stack in zwei Teile teilen und beide Teile zurückgeben. Das Element `delimiter` dient als Trennungskriterium. Der erste Stack beginnt bei dem ersten Element des ursprünglichen `stack` und endet *vor* dem Element `delimiter`. Der zweite Stack beginnt *nach* dem Element `delimiter` und endet mit dem Ende des ursprünglichen Stacks. Das Element `delimiter` soll also in keiner der beiden neuen Stacks auftreten. Die Reihenfolge der Elemente in beiden Stacks ist zu beachten! Zum Erzeugen eines Templatearrays gibt es die Funktion `Array.newInstance(Class componentType, int length);`, wobei `length` die Länge des Arrays sein soll, und `componentTyp` der Klassentyp der Elemente ist. Haben Sie bereits ein Objekt des gleichen Types, können Sie den Klassentypen mit der Methode `getClass()` erhalten.

42 Zyklische Liste

Eine zyklische Liste ist eine LIFO-Datenstruktur mit fester Länge, dessen Enden miteinander verknüpft sind: $x_0 \to x_1 \to ... \to x_n \to x_0$.

(a) Definieren Sie eine Klasse `Node<T>`, die als Knotenpunkt einer einfach verketteten Liste dienen soll. Dazu muss sie einen Zeiger auf den nächsten Knoten sowie den Wert vom Typ T hinterlegen.

(b) Schreiben Sie die Klasse `CyclicList<T>`, die ein Attribut `anfang` und `ende` als `Node`-Zeiger beherbergt. Dem Konstruktor wird die maximale Anzahl an zu speichernden Elementen übergeben. Anschließend wird in einer Schleife obige Verkettung $x_0 \to ...$ erzeugt.

(c) Die Liste ist leer, falls `anfang == ende`. Ansonsten lässt sich die Anzahl der gespeicherten Elemente als Anzahl der Nodes zwischen `anfang` und `ende` berechnen. Schreiben Sie eine Methode `int size()`, die genau das macht.

(d) Die Methode `void push(T wert)` soll in dem Knoten nach `ende` den Wert `wert` schreiben und `ende` auf sein Nachfolgerelement setzen.

(e) Die Methode `T pop()` liefert den Wert von `anfang`. Anschließend wird `anfang` um eins erhöht. Falls die Liste leer ist, müssen Sie eine geeignete Exception werfen.

Testen Sie die Klasse `CyclicList<Integer>` in der `main`-Routine mit Zufallszahlen.

43 Feldkanone

1805. Sie befinden sich in einem Feldzug Napoleon Bonapartes gegen die österreichische Armee. Für die Schlacht sind Sie dazu beauftragt worden, die Effizienz einer Feldkanone Ihres Regiments zu optimieren. Dazu müssen Sie ein Java-Programm schreiben, welches den Ablauf von Laden und Schießen simuliert.

- Ihre Feldkanone kann nur schießen, wenn sie geladen ist.
- Zum Laden benötigen Sie eine Person, die die Kanone mit einer Kugel belädt - den *Lader*.
- Damit Sie sich die Hände nicht dreckig machen, haben Sie einen zusätzlichen *Schießer* abkommandiert.
- Leider lassen sich nur 10 Kugeln in der Nähe der Kanone lagern. Deswegen haben Sie einen *Träger*, der Ihnen mit dem Schubkarren 8 Kugeln aus dem Vorratszelt holen kann.

(a) Ihre Klasse `Feldkanone` muss abspeichern, ob sie geladen ist, und wieviele Kugeln noch bereitstehen.

(b) Kreiiren Sie jede Person als `Runnable`-Klasse. Bedenken Sie, dass der Lader am wenigsten Zeit, der Träger am meisten Zeit mit seiner Tätigkeit beansprucht.

(c) Der Lader muss natürlich solange warten, bis der Schießer die Kugel (ver/ge)schossen hat. Der Träger kann nur seinen Schubkarren entleeren, wenn auf dem Schlachtfeld noch Platz ist.

44 JavaDoc

Erweitern Sie einer Ihrer Lieblingsaufgaben mit JavaDoc-Kommentaren. Verwenden Sie möglichst viele der in der Vorlesung behandelten Schlüsselwörter, die Sie in den Kommentaren einbauen können. Generieren Sie anschließend die Dokumentation im HTML-Format und vergewissern Sie sich über die Korrektheit der Daten, die diese Dokumentation bereitstellt.

45 TicTacToe II

Programmieren Sie nun das TicTacToe Spiel als GUI-Anwendung. Das Anwendungsfenster soll ein 3x3 Feld aus `JButton`s beinhalten, das das Spielfeld repräsentiert. Zusätzlich beinhaltet es einen `JButton`, der die Anwendung beendet, wenn der Button gedrückt wurde. Für die Ergebnisanzeige verwenden Sie einen `JLabel`, wobei Sie für den Label sowohl als auch für den Button die Methoden `getText()` und `setText(String)` verwenden können. Für die Interaktion mit dem Button-Feld brauchen Sie einen ActionListener. Diesen können Sie als innere Klasse implementieren.