Übungen zur Vorlesung
„Praktische Informatik 2“

AG Betriebssysteme, FB 3
Jan Bredereke

SoSe 2002

ausgegeben: 6.5.2002

Blatt 2

Stacks

Aufgabe 1: Nicht-begrenzter Stack (40%)

Implementiert einen Stack, dessen Fassungsvermögen nur durch den verfügbaren Speicher des Rechners begrenzt ist, und zwar mithilfe einer einfachen Liste. Schreibt eine Klasse `UnboundedStack`, die die Methoden `push(obj)`, `pop()`, `peek()` und `isEmpty()` zur Verfügung stellt. Die Funktionalität des Stacks soll die übliche sein, wie auch in der Vorlesung vorgestellt. (Ein leerer Stack soll bei `pop()` und `top()` die Null-Referenz `null` zurückliefern.) Dieser Stack soll zur Vereinfachung nicht beliebige Datenobjekte aufnehmen können, sondern die Elemente auf dem Stack sollen immer vom Typ `StackItem` sein. `StackItem` soll eine Klasse mit drei öffentlichen Feldern sein: `String Artikeltyp`, `int ArtikelId` und `int Artikelvolumen`. Die Klasse `StackItem` soll keine Methoden haben.

Löst das Problem, indem Ihr zuerst eine Klasse `StackList` schreibt, die eine einfache Liste über `StackItem`s implementiert. Die Instanzen dieser Klasse sollen jeweils zwei Komponenten haben, nämlich eine Referenz `obj` auf das zugehörige Objekt im Stack und eine Referenz `prev` auf das vorige Listenelement. Unter Verwendung dieser Listenstruktur implementiert Ihr anschließend die Klasse `UnboundedStack`.

Aufgabe 2: Stapel und Roboter (60%)

Betrachtet das folgende Szenario: In einem kleinen Unternehmen werden aus Kostengründen (Regale sind teuer ...) alle Lagerartikel einfach auf große Stapel gelegt. Um dennoch in der Lage zu sein, einen Artikel halbwegs leicht wiederzufinden, sind diese Stapel sortiert. Dabei sollen die größten Objekte immer unten angeordnet werden (sonst kippt der Stapel zu leicht um). Leider werden im Wareneingang neue Lieferungen immer in unsortierten Stapeln angeliefert. Glücklicherweise hat sich aber in der letzten Woche ein manisch-depressiver Roboter vorgestellt, der eigentlich nur auf das Ende der Welt wartet (nennen wir ihn Marvin). Er hat sich bereit erklärt, nebenbei auch noch (kostenlos) immer die Artikel der unsortierten Eingangsstapel unter Zuhilfenahme eines Hilfsstapels im Lager jeweils auf einem neuen Stapel sortiert aufzuschichten.

Schreibt für Marvin ein Java-Programm, welches ihm sagt, wie seine Sortieraufgabe für einen gegebenen unsortierten Stapel durch hin- und herstapeln erledigt werden kann. Die grundlegende Idee besteht darin, daß der Eingangsstapel schrittweise abgebaut wird und auf den Hilfsstapel umgeschichtet werden kann, wobei man sich in einer internen Variablen (einer `StackItem`-Referenz) merkt, welches bislang das größte Objekt ist. Wenn der Stapel komplett übertragen ist, so kennt man das momentane Maximum und kann es beim Zurückschichten des Hilfsstapels in den Eingangsbereich auf den sortierten Stapel legen. Dies wiederholt man so lange, bis der Eingangsstapel leer ist, der Lagerstapel ist dann sortiert. (Das Verfahren ist nicht das effizienteste, aber Marvin hat genug Zeit ...)

Der Wareneingangsstapel soll zu Anfang aus einer Datei in einen `UnboundedStack` eingelesen werden (siehe Aufgabe 1). Der Name der Datei soll auf der Kommandozeile angegeben werden. In der Datei soll in jeder Zeile der Typ des Artikels (Schraube, Motorblock, Pferd, ...), eine firmeninterne Kennnummer (z.B. 100042) und das Volumen (als ganze Zahl) angegeben werden, durch jeweils ein Blank getrennt. Für das Einlesen aus der Datei benutzt Ihr die Streams `FileReader` und `BufferedReader` (oder nach Eurer Wahl auch `LineNumberReader`). Für die Zerlegung einer Zeile in die drei Blank-getrennten Teile benutzt Ihr den bekannten `StringTokenizer`. Für die Konversion eines Strings in eine ganze Zahl bietet sich die Methode `parseInt(str)` der Klasse `Integer` an. Beachtet, daß die Eingabedatei Fehler enthalten könnte. Insbesondere kann `parseInt(str)` eine Exception werfen, die Ihr fangen müßt.

Der Sortieralgorithmus soll auf dem Bildschirm Anweisungen für Marvin ausgeben, was er tun soll, also z.B. „x mit Id y und Volumen z: Eingangsstapel `->` Hilfsstapel“ (Die Angabe des Objekts ist nur eine Hilfsinformation zum Nachvollziehen des Algorithmus'; durch die Struktur des Stapels ist dieses Objekt bei einer konkreten Stackbelegung immer eindeutig definiert).

Abgabe: 21.–23. Mai 2002 in den Übungen bzw. beim Tutor

Die Abgabe soll sowohl elektronisch (Programm-Quellcode) als auch in gedruckter Form (mit LaTeX gesetzter kommentierter und erläuterter Quellcode) erfolgen. Dabei ist auch auf geeignete Testfälle und deren Dokumentation zu achten!