Übungen zur Vorlesung
Praktische Informatik II
Sommersemester 2001

Serie 1

Hashing und Syntaxüberprüfung

Aufgabe 1: Hashtables und Telefonbücher (50%)

Entwickeln Sie ein Java–Programm zur Verwaltung von Telefonbüchern. Ein Telefonbucheintrag für eine Person besteht hierbei vereinfacht nur aus einem Namen in der Form „`Nachname, Vorname`“ sowie einer zugeordneten Telefonnummer inklusive Vorwahl und eventuellen Trennzeichen (also z. B. `0421/123-4567`). Aus Java–Sicht besteht ein Eintrag damit aus zwei Zeichenketten (Strings) beliebiger Länge. Sie können hierbei davon ausgehen, daß Namen nicht doppelt in dem Telefonbuch vorkommen.
Um bei der potentiell grossen Anzahl von Eintragungen in der Lage zu sein, eine Telefonnummer für einen Namen schnell wiederfinden zu können, soll ein String–Hashing verwendet werden, um in einem Array von Listen von Telefonbucheinträgen die zu durchsuchende Liste schnell bestimmen zu können (vgl. hierzu auch die Beispielimplementierung `MyHashS.java` aus der Vorlesung).
Weiterhin soll es möglich sein, eine Liste aller Einträge nach Namen sortiert auf dem Bildschirm auszugeben. Um dies zu beschleunigen, sollen die Telefonbucheinträge zusätzlich zu der Speicherung im Hash–Array auch noch in einer nach Namen sortierten Liste gespeichert werden. Dabei sollen die Telefonbucheinträge **nicht** doppelt im Speicher gehalten werden, sondern nur die *Referenzen* auf die Einträge sollen in den zwei erwähnten Ordnungen organisiert werden!
Insgesamt sollen die folgenden vier Methoden mit geeignet zu definierenden Signaturen implementiert werden:

`insert` fügt einen neuen Telefonbucheintrag in das Telefonbuch sowie in die alphabetisch sortierte Hilfsliste ein.

`remove` löscht einen Eintrag zu einem gegebenen Namen aus dem Telefonbuch und aus der alphabetisch sortierten Hilfsliste.

`getNumber` sucht zu einem gegebenen Namen über die Hash–Liste die dazugehörende Telefonnummer heraus und liefert diese als Rückgabewert.

`listSorted` gibt alle Namen und dazugehörigen Telefonnummern alphabetisch sortiert auf dem Bildschirm aus.

Zeigen Sie mit geeigneten Testfällen, daß Ihre Methoden die gewünschte Funktionalität besitzen.

Aufgabe 2: Syntaxüberprüfung für PN Terme (50%)

Schreiben Sie ein Java Programm, welches vollständig geklammerte mathematische Terme, gegeben in polnischer Notation (PN), auf ihre syntaktische Korrektheit bezüglich der unten angegebenen Grammatik überprüft. Die Ausdrücke sollen dabei aus positiven Zahlen inklusive der 0, den mathematischen Grundoperationen `+,-,*,/`, runden Klammern `()` sowie Kommas zusammengesetzt sein. Daraus ergibt sich für die Token–Definitionen die folgende lexikalische Grammatik :

```
binop = + | / | *
minop = -
lb    = (
rb    = )
komma = ,
num   = 0 | 1 | 2 | 3 | ...
```

`NUM` soll dabei den natürlichen Zahlen inklusive 0 entsprechen. Der Einfachheit halber sind auch Zahlen mit führenden Nullen zugelassen.
Hierauf aufbauend besteht die PN–Grammatik dann aus vier Regeln:

```
TERM    = binop lb ARGS rb
        | minop MINARG

ARGS    = num komma ARGREST
        | binop lb ARGS rb komma ARGREST
        | minop MINARG komma ARGREST

ARGREST = num | binop lb ARGS rb | minop MINARG

MINARG  = lb ARGS rb | num
```

Ihr Programm soll den zu überprüfenden Ausdruck als eine Kommandozeilenparameter–Zeichenkette übergeben bekommen, d. h. es soll später z. B. in der Form `java SynCheck "+(4,-(8,2))"` aufgerufen werden. Der zu prüfende String befindet sich dann in dem Übergabeparameter der `main`–Methode Ihres Programms, und dort in dem Array Element Nummer 0. Als Ergebnis soll auf dem Bildschirm nur ausgegeben werden, ob es sich bei einem Ausdruck um einen korrekten `TERM` handelt oder nicht; es soll dabei **nicht** auch noch das Ergebnis eines übergebenen Ausdrucks berechnet werden.
Trennen Sie in Ihrem Programm die Ermittlung der Tokens aus dem Eingabestring als eigene Methode ab. Hier ist es hilfreich, den von Java in dem Package `java.util` zur Verfügung gestellten `StringTokenizer` zu verwenden. Zur Erkennung, ob es sich um ein `num`–Token handelt, bieten sich die Funktionen `charAt` der `String`–Klasse sowie `isDigit` aus der `Character`–Klasse an.
Für die Implementierung der eigentlichen Syntaxüberprüfung anhand der PN–Grammatik empfiehlt es sich, für jede Regel eine eigene Methode zu schreiben. Dann spiegelt sich die rekursive Natur der Grammatikregeln direkt in einer entsprechenden Aufrufstruktur Ihrer Methoden wieder.
Zeigen Sie anhand geeigneter Testfälle, daß Ihr Programm korrekte und fehlerhafte PN–Terme unterscheiden kann.

Abgabe: 23.–27. April 2001 in den Tutorien. 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!