Übungen zur Vorlesung
Praktische Informatik II
Sommersemester 2003

Serie 2

Syntaxüberprüfung

Aufgabe 1: Syntaxüberprüfung für Taschenrechner mit polnischer Notation (70%)

Die polnische Notation (PN) ist eine Schreibweise für mathematische Terme, die z.B. von manchen Taschenrechnern als Eingabeform verwendet wird. Der Verknüpfungsoperator steht dabei immer vorne, und es folgen die zu verknüpfenden Zahlen oder Ausdrücke danach. Um die Terme leichter lesbar zu machen, verwenden wir Klammern. Den mathematischen Term „$4 + 8 - 2$" schreibt man z.B. als „`+(4,-(8,2))`".
Schreibt ein Java-Programm, welches vollständig geklammerte mathematische Terme, gegeben in PN, auf ihre syntaktische Korrektheit bezüglich der unten angegebenen Grammatik überprüft.
Euer 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 Eures 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.
Die Ausdrücke sollen 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 = ,
digit = 0 | 1 | 2 | 3 | ... | 9
```

Aufbauend auf der lexikalischen Grammatik besteht die PN-Grammatik aus sechs Regeln:

```
TERM    = BINTERM
        | MINTERM
        | NUM

BINTERM = binop ARGS

MINTERM = minop MINREST

MINREST = ARGS
        | NUM

NUM = digit {digit}

ARGS    = lb TERM komma TERM rb
```

`NUM` soll dabei den natürlichen Zahlen inklusive 0 entsprechen. Der Einfachheit halber sind auch Zahlen mit führenden Nullen zugelassen.
Schreibt ein Programm `SynCheck`, daß seine Eingabe auf korrekte PN-Syntax überprüft.
Trennt in Eurem Programm die Ermittlung der Token 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 `digit`-Token handelt, bieten sich die Funktionen `charAt` der `String`-Klasse sowie `isDigit` aus der `Character`-Klasse an.
Schreibt für jede der sechs Regeln der obigen PN-Grammatik eine eigene Methode zur Überprüfung. Dadurch spiegelt sich die rekursive Natur der Grammatikregeln direkt in einer entsprechenden Aufrufstruktur Eurer Methoden wieder.
Zeigt anhand geeigneter Testfälle, daß Euer Programm korrekte und fehlerhafte PN-Terme unterscheiden kann.

Aufgabe 2: Syntaxüberprüfung für JAVA (30%)

Überprüft anhand der JAVA Grammatik auf `http://home.bredband.no/gaulyk/java/grammar/scjp.html`, ob es sich bei den folgende Zeichenketten jeweils um ein korrekt gebildetes *Statement* handelt:
a) `while ( 3 > 17 ) { ; }`
b) `do = 17`
Dabei soll die genaue Ableitung angegeben werden, bzw. (falls es kein korrektes *Statement* ist) die vollständige Auflistung der fehlschlagenden Pfade.

Abgabe: 26. Mai – 29. Mai 2003 in den Übungen
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!