Technische Universität Darmstadt
Fachbereich Informatik
Prof. Johannes Fürnkranz

Allgemeine Informatik II im SS 2007

Übungsblatt 12

Bearbeitungszeit: 20.07. bis 05.09.2007

Aufgabe 1: Postfix-Ausdrücke und Exceptions

Da wir Sie im Programmierprojekt noch nicht genug mit Postfix-Ausdrücken gequält haben, gibt es hier eine Zugabe. Anders als im Projekt geht es hier aber um die Fehlerbehandlung mittels Exceptions, die Auswertung von Postfix-Ausdrücken ist schon fertig. Laden Sie die Datei **uebung12.zip** von der Übungs-Webseite herunter, entpacken Sie sie und öffnen in BlueJ das Projekt **uebung12**. Schauen Sie sich die Klasse **`Postfix`** gut an.

In der Methode **`int eval(String ausdruck)`** wird ein arithmetischer Ausdruck in Postfix-Schreibweise zu einer Zahl ausgewertet (**`"23 42 + 5 *"`** würde also das Ergebnis **`325`** liefern). Dazu wird der String zuerst in Tokens zerlegt, wenn auch etwas anders als im Projekt: der Umweg über **`Vector`** fällt weg und die **`Iterator`**-Methoden **`hasNext()`** und **`next()`** heißen jetzt **`hasMoreTokens()`** und **`nextToken()`**. Außerdem speichert der **`Stack`**, den wir zur Auswertung brauchen, nur **`Integer`**-Objekte, also Zahlen – das liegt daran, dass wir nur am Ergebnis der Rechnung interessiert sind und keinen Baum erzeugen wollen.

Wenn Ihnen bis hierhin etwas unklar ist, lesen Sie noch mal die Aufgabenstellung des Programmierprojekts!

Im Projekt sollten Sie Fehler einfach ignorieren – jetzt wollen wir aber zumindest einige Fehler abfangen und entsprechende Fehlermeldungen ausgeben. Dafür bieten sich in Java natürlich **`Exceptions`** an!

1. Schreiben Sie eine Exception (also eine Klasse, die von **`Exception`** erbt) namens **`PostfixException`**.

   a. Diese soll einen Konstruktor haben, dem ein Postfix-Ausdruck als **`String`**, eine Fehlerstelle als **`int`** und eine Fehlerart als **`int`** übergeben werden. Initialisieren Sie entsprechende Attribute der Klasse.

   b. Folgende Fehler sollen Sie behandeln (Fehlerart!):
      - Division durch 0
      - das Fehlen eines Operanden zu einem Operator
      - das Fehlen beider Operanden zu einem Operator

   c. Die Methode String **`getMessage()`** Ihrer Exception soll eine aussagekräftige Fehlermeldung als String zurückgeben. Diese soll den Ausdruck und die Fehlerstelle beinhalten.

2. Ändern Sie jetzt die Klasse **`Postfix`** entsprechend ab:

a. An den Stellen in der Methode **`eval()`**, an denen die oben genannten Fehler auftreten können, werfen Sie eine entsprechende **`PostfixException`**. Um die Fehlerstelle herauszufinden, benötigen Sie eine **`int`**-Variable **`stelle`**, die vor der **`while`**-Schleife mit 0 initialisiert und am Ende der **`while`**-Schleife erhöht wird (und zwar nicht einfach um 1, sondern um die Länge des aktuellen Tokens!).

b. Damit Sie die Exceptions nicht auch in der Methode **`eval()`** fangen müssen, ist noch eine kleine Änderung am Methodenkopf nötig.

c. Bleibt noch die Methode **`postfix()`**: fangen Sie mittels **`try`** und **`catch`** auftretende **`PostfixException`**s und geben Sie deren Fehlermeldung aus.

Wenn Sie wollen, können Sie auch versuchen, einen Fehler auszugeben, wenn ein Operator zu viele Operanden bekommt bzw. ein Operator fehlt…

Aufgabe 2: I/O

Erstellen Sie im Projekt aus Aufgabe 1 eine neue Klasse **`IO`**. Importieren Sie die Klassen des Package java.io („**`import java.io.*;`**“). Schreiben Sie nun eine Methode **`public static void io()`**, die aus einer Datei zeilenweise Daten liest und diese durch die Klasse **`Postfix`** als Postfix-Ausdrücke interpretieren lässt.

Konkret heißt das:

- Sie müssen einen **`FileReader`** erzeugen, der die Datei „**README.TXT**“ öffnet (die Großschreibung ist wichtig).
- Sie müssen einen **`BufferedReader`** erzeugen, der den FileReader übergeben bekommt.
- Der BufferedReader bietet eine Methode **`readLine()`**, um ganze Zeilen einzulesen - nutzen Sie diese, um so lange eine Zeile nach der anderen aus der Datei zu lesen, bis das Ende der Datei erreicht ist (sehen Sie in der **Java-API** nach, wie Sie das herausfinden).
- Übergeben Sie in Ihrer Schleife die jeweils gerade gelesene und in einer **`String`**-Variable zwischengespeicherte Zeile der Methode **`Postfix.postfix(…)`**.
- Schließen Sie anschließend die Datei.
- Beim Öffnen der Datei und beim Lesen aus der Datei können Fehler auftreten – benutzen Sie also **`try`** und **`catch`**, um die beiden möglichen Exceptions zu fangen und entsprechende Fehlermeldungen auszugeben.

Die Datei README.TXT können Sie direkt in BlueJ öffnen, indem Sie auf das weiße Symbol links oben im Klassenstrukturfenster doppelklicken. Zu Beginn stehen dort einige korrekte und fehlerhafte Postfix-Ausdrücke. Schreiben Sie aber auch gerne eigene…

Viel Spaß und viel Erfolg in der Klausur!

Wenn Sie Fragen zu dieser oder anderen Übungen haben, steht Ihnen das Forum natürlich auch während der vorlesungsfreien Zeit zur Verfügung.