Institut für Systemsoftware
Prof. Dr. H. Mössenböck

Übungen zu Übersetzerbau WS 2007/2008

Übung 2

Zuname _____ Vorname _____ Matr.-Nr. _____

Übungsgruppe

- ❑ 1 (Löberbauer) Do 10^{15}-11^{45}
- ❑ 2 (Löberbauer) Do 13^{45}-15^{15}
- ❑ 3 (Wimmer) Do 10^{15}-11^{45}

Punkte _____ korr. _____

Letzter Abgabetermin
Mittwoch, 24.10.2007, 20^{15} Uhr

Lexikalischer Analysator (Scanner) (24 Punkte)

Beginnen Sie die Arbeit an Ihrem *MicroJava*-Compiler, indem Sie einen lexikalischen Analysator (*Scanner*) für die Sprache *MicroJava* schreiben. Lesen Sie dazu die Sprachdefinition (VO-Skriptum, Kapitel 4, Seiten 6ff) genau durch. Welche Terminalsymbole und Terminalklassen werden in der Grammatik gebraucht? Wie sehen die Kommentare aus, wie die Zeichenkonstanten? Welche Schlüsselwörter und vordeklarierten Namen gibt es?

Auf der Homepage finden Sie vorgegebene Klassen für die Implementierung. Viele Klassen sind für diese Übung noch nicht nötig, sondern werden erst im Laufe der nächsten Übungen implementiert. Für diese Übung benötigen Sie die Klassen *Scanner*, *Token* und *Errors* im Package *ssw.mj*. Die Klassen *Token* und *Errors* sind bereits fertig implementiert, bei der Klasse *Scanner* sind einige Definitionen vorgegeben. Diese Klasse muss vervollständigt werden.

Bei jedem Aufruf von *Scanner.next()* muss das nächste erkannte Terminalsymbol (= *Token*) zurückgegeben werden. Wenn das Ende des Eingabestroms erreicht ist, wird das Terminalsymbol *eof* zurückgegeben.

Schlüsselwörter, Bezeichner, (positive ganze) Zahlen, Zeichenkonstanten sowie Operatoren müssen erkannt werden. Leerzeichen, Tabulatoren, Zeilenumbrüche und Kommentare (von "/*" bis "*/", auch geschachtelt) müssen überlesen werden.

Folgende lexikalische Fehler müssen erkannt werden:

- Das Auftreten ungültiger Zeichen (*INVALID_CHAR*).
- Fehlende schließende Anführungszeichen bei Zeichenkonstanten (*MISSING_QUOTE*).
- Leere Zeichenkonstanten (*EMPTY_CHARCONST*)
- Ungültige Escapesequenzen (*UNDEFINED_ESCAPE*).
- Zeilenumbrüche in Zeichenkonstanten (*ILLEGAL_LINE_END*)
- Nicht geschlossene Kommentare (*EOF_IN_COMMENT*)
- Zu große Zahlenkonstanten (*BIG_NUM*)

Der Wertebereich von *int* ist wie in Java von *-2147483648* bis *2147483647* definiert. Der lexikalische Analysator liefert jedoch nur positive ganze Zahlen (also von *0* bis *2147483647*). Bei einer negativen Zahl werden ein Minuszeichen und eine positive ganze Zahl als zwei separate Tokens geliefert. Die Zahl *-2147483648* kann daher nicht als Konstante im Programm direkt angegeben werden.

Im Vorlesungs-Skriptum sind alle Methoden der Klasse *Scanner* als *static* deklariert. Im Gegensatz dazu verwenden wir in der Übung keine statischen Methoden, um die JUnit-Testfälle zu vereinfachen. Daher müssen Objekte der Klasse *Scanner* und *Errors* angelegt werden. Die Initialisierung findet in den jeweiligen Konstruktoren statt.

JUnit-Tests

Implementieren Ihren Compiler so, dass er alle vorgegebenen JUnit-Testfälle besteht. Auch die Tutoren verwenden diese Testfälle. Die Übungen werden nur beurteilt, wenn alle Testfälle ohne JUnit-Errors durchlaufen.

Sie können die gegebenen JUnit-Testfälle auch als Vorlage für Ihre eigenen Tests verwenden, um so Ihren Compiler noch ausführlicher zu testen.

Abgabe

Die Abgabe der Übungen 2 – 6 muss auf Papier und elektronisch erfolgen. Geben Sie folgende Dateien ab:

- Ausgedruckt auf Papier: *Scanner.java* (andere Dateien müssen für diese Übung nicht verändert werden).
- Elektronisch als ZIP-Datei: **Alle** Quellcode-Dateien, die zum **Ausführen** des Compilers benötigt werden (Packages *ssw.mj*, *ssw.mj.codegen und ssw.mj.symtab*), also auch alle Klassen der Angabe. Die Verzeichnis-Struktur muss in der ZIP-Datei erhalten bleiben.
- **Nicht abzugeben:** JUnit-Testfälle, *.class*-Dateien, Projekt-Dateien von IDEs.