

Zuname _____ Vorname _____ Matr.-Nr. _____

Übungsgruppe

- | | | |
|--------------------------|----------------|---------------------------------------|
| <input type="checkbox"/> | 1 (Löberbauer) | Do 10 ¹⁵ -11 ⁴⁵ |
| <input type="checkbox"/> | 2 (Löberbauer) | Do 13 ⁴⁵ -15 ¹⁵ |
| <input type="checkbox"/> | 3 (Würthinger) | Do 10 ¹⁵ -11 ⁴⁵ |

Punkte _____ korr. _____

Letzter Abgabetermin

Mittwoch, 28.10.2009, 18⁰⁰ Uhr

Lexikalischer Analysator (Scanner)

(24 Punkte)

Schreiben Sie einen lexikalischen Analysator (*Scanner*) für die Sprache *MicroJava*. Die lexikalische Struktur vom *MicroJava* finden Sie im VO-Skriptum in Kapitel 4.2 auf Seite 7.

In ihrem Repository finden Sie vorgegebene Klassen für die Implementierung. Für diese Übung brauchen Sie die Klassen *Scanner*, *Token* und *Errors* im Package *ssw.mj*. Die Klassen *Token* und *Errors* sind bereits fertig implementiert. Die Klasse *Scanner* müssen Sie für diese Übung vervollständigen.

Scanner.next() muss das jeweils nächste erkannte Terminalsymbol (*Token*) zurückgeben. Wenn das Ende des Eingabestroms erreicht ist, muss *eof* zurückgegeben werden.

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*)
- Ungeschlossene Kommentare (*EOF_IN_COMMENT*)
- Zu große Zahlenkonstanten (*BIG_NUM*)

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

Im Vorlesungs-Skriptum sind alle Methoden der Klasse *Scanner* *static* deklariert. In der Übung verwenden wir dynamische 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 aus *ScannerTest* besteht. Auch die Tutoren verwenden diese Testfälle.

Sie können die gegebenen JUnit-Testfälle auch als Vorlage für eigene 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* (nur diese Datei muss für diese Übung verändert werden).
- Elektronisch in das Repository: **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 Abgabe. Die Verzeichnis-Struktur muss erhalten bleiben.
- Taggen sie die Abgabe als: Abgabe UE1