

Zuname _____

Matr. Nr. _____

Übungsgruppe:

Punkte _____ korr. _____

- 1 (Wöß) Do 10¹⁵ - 11⁴⁵
- 2 (Wöß) Do 12⁰⁰ - 13³⁰
- 3 (Rammerstorfer) Do 17¹⁵ - 18⁴⁵

Letzter Abgabetermin:

Donnerstag, 30.10.2002, 8¹⁵ Uhr

Lexikalischer Analysator (*Scanner*)

a) Implementierung

(20 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-Skript Seiten 8-13) 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?

Folgende Schnittstellen sind gegeben:

(Sie finden die Dateien auf der Homepage (UB-UE2-Angabe.zip))

Datei *Token.java*:

```
package ssw.mj;
public class Token {
    // ----- token codes
    public static final int
        none = 0, ident = 1,
        number = 2, charConst = 3, ...,
        ifKW = 34, // append 'KW', because 'if' is Java keyword
        newKW = 35, ..., eof = 41;

    int kind; // token kind
    int line; // token line
    int col; // token column
    int val; // token value for number & charConst
    String string; // string representation of token
}
```

Datei *Scanner.java*:

```
package ssw.mj;
public class Scanner {
    //----- static variables
    private static Reader in; // input stream (source file)
    private static PrintWriter out; // output stream
    private static char ch; // lookahead character
    private static int line; // current line
    private static int col; // current column

    //----- print error message to out
    private static void error (String msg) { ... }
    //----- initialize scanner
    public static void init (Reader r) { ... }
    //----- return next input token
    public static Token next () { ... }
}
```

Beachten Sie:

- Bei jedem Aufruf von *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 und Zeilenumbrüche müssen überlesen werden.
- Kommentare (von `"/**" bis "*/"` auch geschachtelt) müssen überlesen werden.
- Folgende Ereignisse haben einen lexikalischen Fehler zur Folge:
 - Das Auftreten ungültiger Zeichen (`INVALID_CHAR`, `INVALID_CHARCONST`, `UNDEF_ESC`).
 - Fehlende schließende Anführungszeichen bei Zeichenkonstanten (`MISS_QUOTE`).
 - Leere Zeichenkonstanten (`EMPTY_CHARCONST`).
 - Zu große Zahlenkonstanten (`BIG_NUM`): der Wertebereich von *int* ist wie in *Java* von `-2147483648` bis `2147483647` definiert.
Achtung: der lexikalische Analysator muss nur positive ganze Zahlen liefern (also von 0 bis `2147483647`). Negative Zahlen werden erkannt, indem ein Minuszeichen und eine positive ganze Zahl als zwei separate Tokens bei wiederholtem Aufruf von *next()* geliefert werden. (Anm.: `-2147483648` kann daher nicht als Konstante im Programm direkt angegeben werden. Will man diese Zahl tatsächlich verwenden, müsste man `-2147483647-1` schreiben).

Lexikalische Fehler sollen Fehlermeldungen (siehe Klasse `ErrorStrings`) nach sich ziehen. Geben Sie diese unter Angabe von Zeilen- und Spaltenposition der Fehlerstelle aus.

Beispiel: `-- line 10, col 8: invalid character '€'!`

b) Test

(2 Punkte)

Implementieren Sie alle Teile Ihres Compilers so, dass er mit den vorgegebenen JUnit-Testfällen getestet werden kann. Auch die Tutoren verwenden diese (und ev. weitere) Testfälle. D.h. die Übungen werden nur beurteilt, wenn die Testfälle ausführbar sind.

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

Testen Sie auf jeden Fall alle in der Spezifikation angegebenen Eigenschaften.

Testen Sie auch die Reaktionen bei fehlerhaften Eingaben!

Richtlinien und Anleitungen zum Testen finden Sie auf der UE-Homepage.

c) Abgabe

(2 Punkte)

Die Abgabe der Übungen 2-6 soll auf Papier **UND** elektronisch erfolgen.

Um das elektronische Abgabesystem zu testen, gibt es für diese Aufgabe alleine schon Punkte! Probieren Sie die elektronische Abgabe aus, indem Sie eine Datei `test.zip`

bis spätestens **Mittwoch, den 22. Oktober, 2003, 12:00 (mittags)**

"abgeben" (diese muss nichts Sinnvolles enthalten und möglichst unter 1KB groß sein).

Richtlinien und Anleitungen für die Abgabe finden Sie auf der UE-Homepage.