Inst. f. Praktische Informatik, Abt. SSW **Übungen zu Übersetzerbau WS 2003/2004**

Prof. Dr. H. Mössenböck **Übung 3**

Name _____ Matr. Nr. _____

Übungsgruppe: Punkte _____ korr. _____

- ❑ 1 (Wöß) Do 10^{15} - 11^{45}
- ❑ 2 (Wöß) Do 12^{00} - 13^{30} Letzter Abgabetermin:
- ❑ 3 (Rammerstorfer) Do 17^{15} - 18^{45} Donnerstag, 13.11.2003, 8^{15} Uhr

---

Syntaxanalysator

a) Rekursiver Abstieg (24 Punkte)

Implementieren Sie den Syntaxanalysator *Parser.java* für *MicroJava* im rekursiven Abstieg. Jede Regel der Grammatik soll dabei durch eine eigene Methode vertreten sein, welche die Top-Down-Erkennung realisiert.

Die Schnittstelle des Parsers nach außen ist durch die Methode *parse* definiert, mit der man die Analyse startet, wobei bei der parameterlosen Version die Ausgabe auf *java.lang.System.out* gelenkt werden soll, während die zweite Variante die Ausgabe auf den gegebenen *java.io.PrintWriter* umleitet (Dies benötigen wir wieder für unsere JUnit-Tests).

Benutzen Sie außerdem die drei in der Vorlesung vorgestellten Methoden *scan*, *check* und *synError*. Bei der Fehlerbehandlung verwenden Sie (vorläufig) die „*Panic Mode*"-Strategie, d.h. Sie brechen die Syntaxanalyse beim ersten Fehler ab (nachdem Sie eine entsprechende Fehlermeldung ausgegeben haben).

Verwenden Sie für Ihren Parser folgende Schnittstelle:

```
package ssw.mj;

public class Parser {
  private static Token t;          // last recognized token
  private static Token la;         // look ahead token (not yet recognized)
  private static int sym;          // always holds la.kind
  private static PrintWriter out;  // all compiler output goes here

  private static void scan () {…}                // reads one symbol ahead
  private static void check (int expected) {…}   // verifies symbol and reads ahead
  private static void synError (String msg) {…}  // prints an error message to out

  private static void Program () {…}             // recognizes NTS 'Program'
  private static void ConstDecl () {…}           // recognizes NTS 'ConstDecl'
  ...
  private static void Mulop () {…}               // recognizes NTS 'Mulop'

  public static void parse () {…}                 // starts the analysis
  public static void parse (PrintWriter output) {…}      // starts the analysis
}
```

Codegerüste, Compilerprogramm und MicroJava-Testprogramme sowie JUnit-Tests finden Sie auf der Homepage.