

**Praktikum 1 zu TI****22.05.2019 u. 29.05.2019****SoSe 19**

---

**Ziele:**

Ziel des ersten Praktikumversuches ist die Entwicklung eines Parsers für WHILE0-Programme mittels JavaCC.

Wie in Übungsaufgabe 13 sind WHILE0-Programme WHILE-Programme (Definition siehe Skript), in denen Wertzuweisungen auf solche der Form

$V1 = V2 + 1$  und  $V1 = 0$  beschränkt sind

und weitere Anweisungen nur durch Sequenz  $\alpha 1; \alpha 2$

und Wiederholungsanweisungen der Form `while V1 != V2 do begin  $\alpha$  end` konstruiert sind.

**Aufgabe 1:**

Geben Sie ein Syntaxdiagramm zur Beschreibung der Syntax von WHILE0-Programmen an.

Gehen Sie für die Entwicklung des Parsers wie folgt vor:

- 1) Als Entwicklungsumgebung *können* Sie z.B. Eclipse mit dem JavaCC-Plugin verwenden. Sie finden auch im Skript weitere Informationen über JavaCC. Es wird empfohlen im Default-Package zu arbeiten.
- 2) Erstellen Sie eine .JJ-Datei, die den Scanner (Tokendefinition als Reguläre Ausdrücke) und Parser (Typ 2-Grammatik in JavaCC-Notation) enthält.
  - (Offizielle) Dokumentation zu JavaCC: <https://javacc.org/doc>
  - In der .JJ-Datei kann die Syntax der While0-Sprache mittels EBNF-ähnlichen Regeln definiert und mit beliebigem Java-Code erweitert werden
  - Definieren Sie sinnvolle Token als reguläre Ausdrücke zur Beschreibung der lexikalischen Konstrukte, z.B. NUMBER, IDENT usw.

**Codebeispiel für Token** (Vorsicht: nicht vollständig und konsistent zu WHILE0, dient lediglich als weiteres Beispiel!):

Achten Sie hierbei darauf, dass die Token für die Schlüsselwörter vor den anderen aufgeführt werden, da dadurch Konflikte aufgelöst werden.

TOKEN:

```
{
  < WHILE: "while">
  |
  < NUMBER : "0" | (["1"-"9"] (["0"-"9"])* ) >
  |
  < IDENT : (["a"-"z", "A"-"Z", "0"-"9"])+ >
  |
  < NOTEQUAL: "!=">
  |
  < ASSIGN: "=">
}
```

- Erstellen Sie die Regeln für die Syntax der WHILE0-Sprache. In JavaCC werden Regeln als Methoden implementiert (in diesem Versuch ist es noch nicht zwingend notwendig diese Regeln mit Java-Code zu erweitern)

**Praktikum 1 zu TI****22.05.2019 u. 29.05.2019****SoSe 19**

---

**Codebeispiel für Regeln:**

```

void programm() throws IdentException:
{
    // In diesem Block kann beliebiger Java-Code stehen
    String s="";
}
{
    // In diesem Block stehen Token, Terminalsymbole oder weitere
    // Methodenaufrufe (auch mit Parametern möglich), reguläre
    // Ausdrücke sind hier auch erlaubt
    <IDENT> "(" inputVars() ";" outputVars() ")" ";"
    vars() ";"
    p()
    <EOF> //Kennzeichnet das Ende, wird durch STRG-Z ausgelöst
}

void inputVars() throws IdentException:
{}
{
    // hier Code rein für inputVars-Zweig
}

```

- Im letzten Schritt muss in der .JJ-Datei ein Hauptprogramm geschrieben werden, das ein WHILE0-Programm einliest und dazu eine Ausgabe liefert
- Im ersten Praktikum soll die Ausgabe nur beinhalten, ob das eingegebene Programm ein gültiges While0-Programm ist oder nicht (im zweiten Praktikum liefert die Ausgabe dann den in URM übersetzten Quellcode)

**Aufgabe 2:**

Überlegen Sie sich, wie Sie den WHILE0-Befehl

```
while V1 != V2 do begin α end
```

in *gültige* URM-Befehle übersetzen können.