Praktikum 3 Compilerbau WS14/15 Testat bis 18.11.2014

Ziele:
- Erweiterung der Postfixtransformation aus Versuch 2 um Konstantendefinitionen.
- 1. Schritt der Implementierung einer Symboltabelle in Java
- Implementierung des Parsers für Mini-Java.

Aufgabe 1: (Konstantendefinition & Symboltabelle)
Erweitern Sie Ihren Postfixtransformator aus Versuch 2, so dass der Ausdruck jetzt auch Bezeichner enthalten kann, die in einer Konstantendefinition, wie in der Syntax von Mini-Java beschrieben, definiert werden.
Programme können jetzt bspw. wie folgt aussehen:

```
final int i = 5, j = 7;
i+3*j
```

Dabei sollen die Initialisierungen der Konstanten im Deklarationsteil in eine als Java-Klasse zu implementierende Symboltabelle eingetragen werden und die Konstanten in der Postfixdarstellung durch ihren Wert ersetzt werden.

Als Ausgabe zum obigen Beispielprogramm ergibt sich dann: `537*+`
Hierzu ist die Konfigurationsdatei von JavaCC u.a. so zu erweitern, dass Bezeichner und Zahlen erkannt und unterschieden werden.

Lösen Sie in den Fällen, dass eine Konstante mehrmals definiert wird oder im Ausdruck verwendet wird, ohne dass sie vorher definiert wurde, selbst implementierte Exceptions aus!
Außer den Klassen zur Implementierung der Symboltabelle und der Exceptions sollen alle anderen Veränderungen direkt in der JavaCC-Konfigurationsdatei realisiert werden!

Hinweis: Stellen Sie die Funktionalität einer Symboltabelle durch eine selbst zu implementierende Java-Klasse zur Verfügung. Hierzu können Sie eine frei zu wählende Datenstruktur verwenden.
Es müssen jedoch folgende Methoden vorhanden sein:

- `void addConstant (String id, String wert)`, die eine Konstante mit deren Wert in die Symboltabelleeinträgt, falls die Konstante noch nicht vorhanden ist, und sonst eine selbst zu definierende `SymbolAlreadyDefinedException` wirft.
- `String getSymbol (String id)`, die den Wert der Konstanten aus der Symboltabelle liefert, falls die Konstante vorhanden ist, und sonst eine selbst zu definierende `UnknownSymbolException` wirft.

Aufgabe 2: (Parser für Mini-Java)
Schreiben Sie einen Parser, der genau die Programme von Mini-Java erkennt und im Fall eines gültigen Mini-Java-Programms die Ausgabe `"Programm syntaktisch ok!"` liefert.
Verwenden Sie hierzu die LL(1)-Grammatik für Mini-Java aus Übungsaufgabe 30!
Hinweis: Für den Parser ist keine Symboltabelle zu verwenden, da er lediglich die Syntaxanalyse durchführt.
Halten Sie als Test das Programm aus Versuch 1 Aufgabe 1 bereit.