

CS126: Introduction to Compilers

Assignment 3: SYN0

Out: 9/19/03

Due: 9/26/03

1.0 Introduction

Syntactic analysis in a compiler is generally a lot more involved than lexical analysis. In this class we will break the generation of an appropriate syntax analysis module for DECAF into three separate assignments. This, the first, will involve writing an input file for JavaCC that has no conflicts but that still accurately reflects the underlying language. The second will involve adding error recovery and error messages. The third will involve adding actions to the grammar in order to generate appropriate abstract syntax trees.

2.0 Specifications

Your task this week is to expand your JavaCC file to include the syntax definitions for DECAF. You should also modify the dummy `parseFile()` method that you created for lexical analysis so that it calls the nonterminal representing the whole program and catches the exception `ParseException` if there is a parsing error.

The grammar for DECAF is given in the handout “Syntax of the Brown Decaf Programming Language” that was distributed during the first week of class. You should be as true to that grammar as possible in creating your parser.

3.0 Difficulties

This assignment is not as easy as it looks. While you might start with a lot of simple typing to convert the grammar rules from the syntax in the handout into the syntax provided by JavaCC, you will soon find out that this alone is not sufficient. There are several places in which the given grammar is not LL(1). Some of these (such as dangling else and expressions) are discussed briefly in the handout. Others you will find are just there and produce various conflicts when you run `javacc`.

You are going to have to take each of these conflicts, determine what causes it, and then either modify your grammar or add appropriate LOOKAHEAD specifications so that it is eliminated. If at all possible (and it is possible), you should avoid requiring later semantic checks or processing for a successful parse. You should also try to avoid having to augment the lexical analyzer.

4.0 Mechanics

Handins should again be done electronically and you can again work either individually or in small teams. If you run the decaf compiler with the `-p` option, it will just run the parser on the specified input file. If you set the option `DEBUG_PARSER=true` in the options section of the JavaCC specification, then you can should call `disable_tracing()` in your constructor to turn tracing off; of course, if you omit this call, then you will get good debugger output to tell you if your parser is working.