Programming Assignment 3
Due April 29th, 2017 at 11:59 PM

1 Introduction

In this assignment you will write a parser for Cool. The assignment makes use of the parser generator (called ANTLR). After completion of this assignment you will be able to classify syntactically incorrect `cool programs`.

You certainly will need to refer to the syntactic structure of Cool, found in Figure 1 of The Cool Reference Manual (as well as other parts). The documentation for ANTLR is available online.

2 Files and Directories

To get started, create a directory where you want to do the assignment and extract parser.tar.gz in it. The files that you will need to modify are:

- src/grammar/CoolParser.g4

  This file contains a start towards a parser description for Cool. The rule section is incomplete. We have provided some parts of a rule. You should not need to modify this code to get a working solution, but you are welcome to if you like. However, do not assume that any particular rule is complete.

- src/java/cool/AST.java
  This file contains the classes for each type of node in the AST. You will not need to modify this file to complete the assignment.

- src/test_case/*.cl
  Make files that test a few features of the grammar. You should add legal tests to ensure every legal construction of the grammar and bad tests that exercise as many types of parsing errors. Explain your tests in these files and put any overall comments in the README file.

- README
  As usual, this file will contain the write-up for your assignment. Explain your design decisions, your test cases, and why you believe your program is correct and robust. It is part of the assignment to explain things in text, as well as to comment your code.

  We will use Google classroom for submissions.

3 Testing the Parser

To test the parser, you will need a working scanner which is already provided.

You will run your parser using parser, a shell script that will parse the input file and print the error messages if any.

You should test this compiler on both good and bad inputs to see if everything is working. Remember, bugs in your parser may manifest themselves anywhere.

4 Parser Output

For programs that have errors, the output is the error messages of the parser. ANTLR has an inbuilt error reporting routine that prints error messages in a standard format; please do not print custom error messages.

For syntactically valid input programs your parser should terminate normally.

Your parser need only work for programs contained in a single file—don't worry about compiling multiple files.

5 Error Handling

Do not be overly concerned about the the line numbers that appear in the error messages your parser generates. If your parser is working correctly, the line number will generally be the line where the error occurred. For erroneous constructs broken across multiple lines, the line number should be the first line of the construct.

For evaluation we will only check the first error that is printed so ensure that it matches with the output of the cool compiler.