Chris Altonji
Allen Chen
Andrew Smith

CS 4240 Phase I

Compiler repository: https://github.com/ClysmiC/tiger-compiler/tree/master
(You can find google-doc links to our table spreadsheets in our github's readme. They are much more readable than opening the raw .csv files in a spreadsheet program)

Project Structure

The project directory contains the following:

- `src/` : The directory containing all of the source code.
  - `com/tiger/compiler` : Source code for the compiler as a whole.
    - `frontend/` : Source code for the front end of the compiler.
      - `scanner/` : Source code for the scanner.
      - `parser/` : Source code for the parser.
- `res/` : The directory containing .csv files of the various tables needed to drive our scanner and parser.
- `test/` : The directory containing all of the test input programs.
  - test_output/ : Sample output for running the scanner/parser on each of the test programs with the -d flag.
- `handwritten/` : hand-drawn DFA's for the scanner. (Note: table form of DFA can be found in `res/`).
- `grammar.txt` : The rewritten grammar, in it's entirety.
- `grammar-rewrite-process.txt` : Text document showing the step-by-step changes we made when performing the grammar rewrite.
- `compile-script.sh` : Simple script to compile all of the .java source files on Linux.
- `compile-script.bat` : Simple script to compile all of the .java files source on Windows.

Compiling and Running

There are a few ways to compile the project. Because of the non-trivial structure of the project directory, the easiest way is to load the project into an IDE such as Eclipse or IntelliJ, and compile using the IDE. However, a few simple scripts are included to make compiling from the command line possible, and relatively easy.

First, open the command line and navigate to the project root directory.

- To compile on Linux, run `compile-script.sh`
- To compile on Windows, run `compile-script.bat`

Run by typing the following command:

- `java -cp src com.tiger.compiler.TigerCompiler <test-file> [-d]`

The path to the test-file should be relative to the root directory (e.g., `test/test1.tiger`). The `-d` argument is an optional debug flag. Without the debug flag, the compiler will either output “Parse successful”, or it will output any errors detected. With the debug flag, it will also output each token type as it scans it.