Safi Nassar - assignment p2

Usage:

- `make clean && make testclean && make && make test`
- `make test`

Executing the tests will generate output files which will be compared to expected result files.

Prerequisites:

- OpenJDK 11

Tests documentation:

`P1 class`

tests the scanner generated for the C– language. testing all tokens types, input that causes errors, character numbers, values associated with tokens, etc. The scanner retrieves the token name, # line, # character, actual value

`void main(String[] args)`

test driver: calls unit test methods

`void testAllTokens()`

test valid and invalid tokens.

- valid tokens are tested by running input file which contains contains all tokens one per line or multiple per line. The scanner correctness is verified by comparing the input and output files.
- Invalid tokens are tested by the same process only that these are compared to an expected errors file.

`void lexer(FileReader in, PrintWriter out)`

runs scanner - pipes input strings through scanner to an output file, writing the analyzed tokens.

`String formatToken(Symbol token)`

create a proper format for the token properties used in the output test.

`String getTokenValue(Symbol token)`

retrieve the appropriate token value corresponding to a symbol.

`config helper class`

configuration class for controling testing bahavior

`pipe helper class`

manages input and output stream for each test instance, creating temporary files for each test instance

---

JLex scanner for C– language

- Source Language: C– (made up language) → Host language: Java → Target language: MIPS
- C– is a simple programming language that uses Pascal identifiers, a small subset of the C++ language.
- Assumption: the compiler lexical and syntax analysis phases would be executed in parallel, where the scanner would stream tokens to the parser for evaluation. The compiler would insert & remove current scopes from the list while the analysis process is being executed, and priority is given to the current scope (latest pushed to the list).

Assignment:

Tasks:

- Writing the scanner in abstract level: cminusminus.jlex specification.
- Writing the main program to test the generated lexer: P2.java.

Submission:

- Create pdf from markdown: `pandoc README.md -o <lastname.firstname.Pn.pdf>`
  - generate markdown from javadoc and remove redundant comments or
  - generate javadoc to extract method headers: `find . -type f -name "*.java" | xargs javadoc -d ../javadoc`
- Add headers for each file
- Verify code format
- Verify code execution on CSL machines