Milica Andric

CS536

P2

2/22/2021

Contents:

The andric.milica.P2 zip file should contain the following:

files/
+---+ allSymbols_expected.out

+---+ allSymbols.in

+---+ allTokens.in

+---+ anyIdentifiers_expected.out

+---+ anyIdentifiers.in

+---+ cminusminus.jlex

+---+ deps/

+---+ eof.txt

+---+ ErrMsg.java

+---+ integerLiterals_expected.out

+---+ integerLiterals.in

+---+ Makefile

+---+ P2.java

+---+ stringLiterals_expected.out

+---+ stringLiterals.in

+---+ sym.java

+---+ andric.milica.P2.pdf

How to build and invoke the program

For convenience, place all project files into one folder. This project requires a Makefile that uses JLex to create a scanner, and also makes the `P2.class`. Run `make` to compile the program. Next, run `make test` to run `P2` and do any needed file comparisons (e.g., using `diff`). It should be clear from what is printed to the console when `make test` is run what errors have been found.

Run `make cleantest` to remove any files that got created by the program when `P2` was run.

Test methods:

The following test methods all open and read from the appropriate .in file, read and write the corresponding variable to an .out file, and verify correctness of the scanner by comparing either the input and output files OR the output and expected output files using the `diff` command.

private static void testIdentifiers() – tests various identifiers

private static void testIntegerLiteral() – tests integer literals

private static void testStringLiterals() – tests string literals

private static void testSymbols() – tests symbols

private static void testAllTokens() – tests all tokens

Expected errors/warnings:

When running `make test`, there are a few expected errors. These 'errors' have been placed in the corresponding testing file (ex: stringLiterals.in contains unterminated string literals, unterminated string literals with bad escaped character, and string literals with bad escaped character) and the appropriate error/warn messages should be printed to the console. There are also a few expected warnings for large integer literals. If the appropriate error/warning messages are printed to the console, this means that the program is running correctly, and the errors/warning are being handled appropriately. Below, I have included a list of the expected warnings/errors.

```
6:1 ***WARNING*** integer literal too large; using max value

7:1 ***WARNING*** integer literal too large; using max value

5:1 ***ERROR*** unterminated string literal ignored "unterminated

6:1 ***ERROR*** unterminated string literal ignored "also unterminated \"

7:1 ***ERROR*** string literal with bad escaped character ignored "backslash
followed by space: \ is not allowed"

8:1 ***ERROR*** unterminated string literal with bad escaped character
ignored "bad escaped character: \a AND not terminated
```

```
9:1 ***ERROR*** unterminated string literal with bad escaped character
ignored "very bad string \
26:1 ***ERROR*** illegal character ignored: ~
27:1 ***ERROR*** illegal character ignored: `
28:1 ***ERROR*** illegal character ignored: @
29:1 ***ERROR*** illegal character ignored: &
30:1 ***ERROR*** illegal character ignored: $
```