Bob Seedorf

WIlliam Clark

3-1-16

Program 4 Deliverable

We have amended our grammar and we are submitting it again to adhere to the parser which we have constructed.

This will represent our new language and we will continue on with this grammar for the duration of the course.

The following is our new grammar:

THIS SECTION DENOTES KEY OR RESERVE WORDS:

**EMPTY\_STRING** = “”

**WSPACE** = " "

**NEWLINE** = “\n”

**LPAREN** = "("

**RPAREN** = ")"

**LCURL** = "{"

**RCURL** = "}"

**LSQUARE** = "["

**RSQUARE** = "]"

**EQUAL** = "="

**PLUS** = "+"

**MINUS** = "-"

**MULT** = "\*"

**DIV** = "/"

**MOD** = "%"

**GTHAN** = ">"

**LTHAN** = "<"

**HAT** = "^"

**BANG** = "!"

**AND** = "&"

**OR** = "|"

**SQUOTE** = "'"

**DQUOTE** = """

**HASH** = "#"

**FSLASH** = "\"

**PERIOD** = "."

**COMMA** = ","

**SEMICOLON** = ";"

**COLON** = ":"

**USCORE** = '\_'

**TILDE** = '~'

**APOST** = '`'

**AT** = '@'

**ATOM** = [A..Z + a..z + 0..9 + USCORE + TILDE + APOST + AT]\*

**KEYWORD** = [“char” + “for” + “while” + “if” + “public” + “private” + “class” + “static” + “void” + “main” + “args” + “String” + “int” + “System.out.println” + “length” + “indexOf” + “charAt”]

**STRING\_LITERAL** = DQUOTE ATOM\*? DQOUTE

**CHAR\_LITERAL** = SQUOTE ATOM\*? SQUOTE

THIS SECTION DETAILS THE GRAMMAR OF OUR LANGUAGE:

**PARAM** = [RPAREN + [ASSIGNMENT + METH\_CALL] RPAREN + [ASSIGNMENT + METH\_CALL] COMMA PAREN]

**WHILE** = LPAREN STATEMENT RPAREN BODY

**FOR** = LPAREN STATEMENT\* RPAREN BODY

**IF** = LPAREN STATEMENT\* RPAREN BODY [ ( [“else if” LPAREN STATEMENT RPAREN BODY]\* “else” BODY? ) + (“else” BODY?) ]?

**METH\_CALL** = PERIOD ATOM LPAREN PARAM RPAREN

**ASSIGNMENT** = [“=” + OPERATOR + ALT\_OP] [EMPTY\_STRING + DQUOTE ATOM DQUOTE + METH\_CALL + ATOM]

**STATEMENT** = ATOM [“while” WHILE + “for” FOR + “if” IF + ATOM ASSIGNMENT + ATOM METH\_CALL] SEMICOLON

**BODY** = LCURL STATEMENT\*? RCURL

**PROGRAM** = CLASS\_DECL

**CLASS\_DECL** = “public” “class” ATOM LCURL MAIN\_METH METH\_DECL\*?

**MAIN\_METH** = “public static void main (String[]” ATOM “)” BODY

**METH\_DECL**= “public” “static” ATOM ATOM LPAREN PARAM\* RPAREN BODY

we have made several changes to accommodate what we have been able to accomplish with our new language.

* implementation of infinite statements inside a for loop’s header
  + our intention is to make every statement except the first and last statement (delimited by semicolons) a condition to be checked by the loop’s condition checking functionality.
* Lack of support for internal classes of the file being examined
  + only one class declaration is allowed per file
* Lack of support for class wide variable definitions
  + variables may only be defined local to methods and the main method
* Every statement must have a semicolon immediately proceeding it

We faced problems, in this language directly related to the capacity of our parser to look ahead. Our language is LL(2), this is because we need to able to pass method calls as parameters to other method calls. This problem requires us to look ahead to see if a basic ATOM, a evaluable STATEMENT, or another method call is being passed to any given method. We were able to overcome this problem by passing, as a parameter in our parser, the name, or ATOM of the first token in the collection of tokens representing one parameter, as the method’s owner, variable name, or first variable in an evaluable statement, to the subsequent corresponding parsing method.

This was the most prominent problem we faced, and I feel it has been one of the most beneficial in learning the techniques of top down parsing. It is not foreseeable how to overcome problems of an LL(n) language, where n is greater than or equal to 3. However, it understandable potential techniques of avoiding such a predicament altogether.

the included sample files demonstrate two functioning files, that will compile successfully, without error, as well as one sample file that will not compile and errors will be noted with print statements explaining problems