Ruby Parser Part 1

For this assignment, we will be writing a parser that is able to parse an input file written in our "**Tiny**" grammar. You should finish writing the "**Parser**" class that I've provided. Your Parser class should **print**:

- Each time it enters or leaves a rule and what the rule is:
  *Entering **[RULE_NAME]** Rule*
  *Exiting **[RULE_NAME]** Rule*
  There are two exceptions: the **ID** and the **INT** rules. In this grammar, **ID** and **INT** are also Token names, so we will not print when we are "entering" or "exiting" those rules since we can just check the token type.
- Each time it recognizes a token and what the token was:
  *Found **[TOKEN_TYPE]** Token: **[LEXEME]***
  For rules with Epsilon definitions, you should also indicate if that definition was chosen:
  *Did not find **[TOKEN_TYPE]** or **[TOKEN_TYPE]** Token, choosing EPSILON production*
- Something, everytime it catches an error:
  *Expected **[TOKEN_TYPE]** found **[LEXEME]***
  When the error could have been multiple things, separate the token types with the word **or**
  *Expected **[TOKEN_TYPE]** or **[TOKEN_TYPE]** or **[TOKEN_TYPE]** found **[LEXEME]***
- How many parse errors were found
  *There were **[X]** parse errors found.*

Parser.rb extends **Lexer.rb** (The lexer that you wrote for your last assignment) and provides a framework for a top-down, recursive-descent parser of the **TINY** language. The parser stays one token ahead in the Token stream **(@lookahead**) and uses the Token to predict how to continue parsing the current instruction and which method to call next.

The **consume()** method calls **nextToken()** in the scanner. The current **@lookahead** Token is discarded, and the next Token in the stream is retrieved. Whitespace Tokens are discarded.

The **match(dtype)** method tries to match the **@lookahead** Token with the provided type (**dtype**). If a match is found, **consume()** is called to retrieve the next Token. Otherwise an error message is displayed and then **consume()** is called to retrieve the next token.

The **program()** method is first called to parse a **TINY** program. Since a **TINY** program consists of a sequence of statements, **program()** calls **statement()** repeatedly until it encounters the **EOF** token.

Complete the parser by providing methods for the appropriate BNF rules in **TINY**.

I have given you my **lexer** and **token** ruby classes. I have also partially written the **parser** for you and have written a **main.rb** file that can run your ruby parser. Your assignment is to **FINISH WRITING THE PARSER** (parser.rb).

I have also included 5 sample input files that you can use to test your program once you've finished writing it.

input[1-3].txt should complete with no parse errors.

input[4-5].txt should have parse errors.

Below are screenshots of what your output should look like for the original (not extra credit) grammar, based on the input files I've provided.

Extra Credit

The extra credit is to implement the Boolean version of the grammar. I will run 5 tests against your code. If you implemented the Boolean version of the parser, each test will be worth 24 points instead of 20 points (still 5 tests). You can test your code by using **input6-8.tiny** that are provided in vocareum.

Ruby Parser Part 1
input3.txt

```
Entering STMT Rule
Entering ASSGN Rule
Found ID Token: x
Found ASSGN Token: =
Entering EXP Rule
Entering TERM Rule
Entering FACTOR Rule
Found LPAREN Token: (
Entering EXP Rule
Entering TERM Rule
Entering FACTOR Rule
Found INT Token: 2
Exiting FACTOR Rule
Entering TTAIL Rule
Did not find MULTOP or DIVOP Token, choosing EPSILON production
Exiting TTAIL Rule
Exiting TERM Rule
Entering ETAIL Rule
Found ADDOP Token: +
Entering TERM Rule
Entering FACTOR Rule
Found INT Token: 3
Exiting FACTOR Rule
Entering TTAIL Rule
Did not find MULTOP or DIVOP Token, choosing EPSILON production
Exiting TTAIL Rule
Exiting TERM Rule
Entering ETAIL Rule
Did not find ADDOP or SUBOP Token, choosing EPSILON production
Exiting ETAIL Rule
Exiting ETAIL Rule
Exiting EXP Rule
Found RPAREN Token: )
Exiting FACTOR Rule
Entering TTAIL Rule
Found MULTOP Token: *
Entering FACTOR Rule
Found ID Token: dog
Exiting FACTOR Rule
Entering TTAIL Rule
Did not find MULTOP or DIVOP Token, choosing EPSILON production
Exiting TTAIL Rule
Exiting TTAIL Rule
Exiting TERM Rule
Entering ETAIL Rule
Did not find ADDOP or SUBOP Token, choosing EPSILON production
Exiting ETAIL Rule
Exiting EXP Rule
Exiting ASSGN Rule
Exiting STMT Rule
There were 0 parse errors found.
```

Ruby Parser Part 1
input4.txt

```
Entering STMT Rule
Entering ASSGN Rule
Found ID Token: x
Expected = found +
Entering EXP Rule
Entering TERM Rule
Entering FACTOR Rule
Found ID Token: y
Exiting FACTOR Rule
Entering TTAIL Rule
Did not find MULTOP or DIVOP Token, choosing EPSILON production
Exiting TTAIL Rule
Exiting TERM Rule
Entering ETAIL Rule
Did not find ADDOP or SUBOP Token, choosing EPSILON production
Exiting ETAIL Rule
Exiting EXP Rule
Exiting ASSGN Rule
Exiting STMT Rule
Entering STMT Rule
Entering ASSGN Rule
Expected id found =
Expected = found 3
Entering EXP Rule
Entering TERM Rule
Entering FACTOR Rule
Expected ( or INT or ID found eof
Exiting FACTOR Rule
Entering TTAIL Rule
Did not find MULTOP or DIVOP Token, choosing EPSILON production
Exiting TTAIL Rule
Exiting TERM Rule
Entering ETAIL Rule
Did not find ADDOP or SUBOP Token, choosing EPSILON production
Exiting ETAIL Rule
Exiting EXP Rule
Exiting ASSGN Rule
Exiting STMT Rule
There were 4 parse errors found.
```

Ruby Parser Part 1
input5.txt

```
Entering STMT Rule
Found PRINT Token: print
Entering EXP Rule
Entering TERM Rule
Entering FACTOR Rule
Found ID Token: x
Exiting FACTOR Rule
Entering TTAIL Rule
Did not find MULTOP or DIVOP Token, choosing EPSILON production
Exiting TTAIL Rule
Exiting TERM Rule
Entering ETAIL Rule
Found ADDOP Token: +
Entering TERM Rule
Entering FACTOR Rule
Found ID Token: y
Exiting FACTOR Rule
Entering TTAIL Rule
Did not find MULTOP or DIVOP Token, choosing EPSILON production
Exiting TTAIL Rule
Exiting TERM Rule
Entering ETAIL Rule
Did not find ADDOP or SUBOP Token, choosing EPSILON production
Exiting ETAIL Rule
Exiting ETAIL Rule
Exiting EXP Rule
Exiting STMT Rule
Entering STMT Rule
Entering ASSGN Rule
Expected id found ?
Expected = found eof
Entering EXP Rule
Entering TERM Rule
Entering FACTOR Rule
Expected ( or INT or ID found eof
Exiting FACTOR Rule
Entering TTAIL Rule
Did not find MULTOP or DIVOP Token, choosing EPSILON production
Exiting TTAIL Rule
Exiting TERM Rule
Entering ETAIL Rule
Did not find ADDOP or SUBOP Token, choosing EPSILON production
Exiting ETAIL Rule
Exiting EXP Rule
Exiting ASSGN Rule
Exiting STMT Rule
There were 3 parse errors found.
```