ICSI 311 Assignment 4 – Expand the Parser

If you have bugs or missing features in this, you will need to fix them before you can continue on to new assignments. This is very typical in software development outside of school.

You must submit .java files. Any other file type will be ignored. Especially ".class" files.

You must not zip or otherwise compress your assignment. Blackboard will allow you to submit multiple files.

You must submit buildable .java files for credit.

You will submit your existing files (Parser.java, Node.java, IntegerNode.java, FloatNode.java, MathOpNode.java,Basic.java, Token.java, Lexer.java) as well as your new files (VariableNode.java, PrintNode.java, AssignmentNode.java,StatementNode.java, StatementsNode.java).

We are going to build on the parser that we created last assignment to add some more useful capabilities and start the process of turning it into a programming language parser.

Create some new AST nodes:

StatementNode – abstract, derives from Node. All statements (like Print, Assignment) should derive from StatementNode.

VariableNode – holds a variable name

PrintNode – holds a list of Node (the things to print)

AssignmentNode – holds a variable node and a Node (which will be the assigned value)

StatementsNode – holds a list of StatementNode (the statements, right now either a print or an assignment)

For all of these, follow the usual rules: private read-only members, appropriate constructor(s), accessors, ToString().

Start by changing Factor() to accept an WORD (i.e. a variable) as well as a number or an expression in parenthesis. When the identifier is found, create a VariableNode and return it.

Parse() right now only works on expressions. We need to introduce more methods to make our parser able to accept a programming language. Change Parse() to call Statements() instead of Expression().

Create a Statements() method; it should accept any number of statements. Statements() should call Statement() until Statement() fails (returns NULL). Create a Statement() method to handle a single statement and return its node or NULL. A single statement, for now, can consist of EITHER a print statement or an assignment. Your Statements() method must take the Node generated by Statement (if it is not NULL) and add it to the Statements AST node.

Create a PrintStatement method which accepts a print statement and creates a PrintNode or returns NULL:

PRINT printList

Create a PrintList method which accepts a print list. A print list consists of a comma separated list of expressions. You will need to add comma to your lexer.

Finally, an assignment statement is of the form:

VARIABLE EQUALS expression

You have already written the Expression parser. Create an Assignment() parser method that accepts an assignment and returns an AssignmentNode or returns NULL. You will need to add the equals sign to parts of your lexer (you already have it for >= and <=).

Rubric	Poor	OK	Good	Great
Code Style	Few comments, bad names (0)	Some good naming, some necessary comments (3)	Mostly good naming, most necessary comments (6)	Good naming, non-trivial methods well commented, static only when necessary, private members (10)
Unit Tests	Don't exist (0)	At least one (3)	Missing tests (6)	All functionality tested (10)
Create the new AST classes	None (0)	Classes missing (5)	All classes present, some methods missing (10)	All classes and methods (20)
Factor accepts variables	None (0)	Attempted (5)		Correct (10)
Statements	None(0)	Attempted (5)		Correct (10)
PrintStatement	None(0)	Attempted (5)		Correct (10)
PrintList	None(0)	Attempted (5)		Correct (10)
Statement	None(0)	Attempted (5)		Correct (10)
Assignment	None(0)	Attempted (5)		Correct (10)