CMSC 401 Spring 2022 - Programming Assignment 3 (HW3):

Max points: 10

Calculator

Problem Description:

Use Hash Tables to build a simple calculator program that supports:

- Integer variables
- Integer constants/values
- three operations: addition, subtraction, multiplication,
- printing out variable's value

The calculator takes instruction as lines from Standard Input (i.e., System.in in java).

Each line is a single instruction. After reading a single instruction, the calculator should interpret it, and execute the required operation. if there is an error in the instruction, the calculator should detect it, print ERROR, and quit. The calculator should be fast. Each instruction should run in expected constant time, by utilizing hash tables. You can assume the number of instructions will be at most 1000.

Each instruction has one of the following 5 forms (varnameL, varnameR1, varnameR2 represent names of some variables, see below):

1. varnameL = integer-value
   e.g. x1 = 123
2. varnameL = varnameR1 + varnameR2
   e.g. x3 = x2 + x1
3. varnameL = varnameR1 – varnameR2
   e.g. x3 = x2 + x1
4. varnameL = varnameR1 * varnameR2
   e.g. x3 = x2 * x2
5. QUIT

The components of each line will be separated by a whitespace, that is, x1=x2+x3 (no spaces) is not legal, it has to be x1 = x2 + x3.

A variable has a name of the form: 'x' followed by between 1 and 8 decimal digits. The first of the digits cannot be 0. For example:

- x1, x100000, x87654321, x123, x2 are valid variable names
- x0, x0001, x, xyz, x1234567890 are invalid

Test cases will always use valid variable names, you do not need to check for that.

A variable stores an integer value (the value will always fit in the Java int type). Each variable has to be assigned a value before it is used on the right-hand side of an instruction. Each variable can only be assigned a value once (see: https://en.wikipedia.org/wiki/Static_single_assignment_form if you are interested why this is beneficial). Trying to assign to a variable for the second time should cause an error. The arithmetic operations given on input to the calculator will never cause overflow for Java’s int type.

After reading each instruction, the calculator should:

1. If the instruction is QUIT, quit the program. Otherwise:
2. Analyze if it is a correct instruction:
    a. Has the variable on the left been assigned a value before? If yes, print “ERROR” and quit
    b. Have all the variables on the right been assigned values before? If no, print “ERROR” and quit
3. If the instruction is correct, the calculator should:
    a. Calculate the value of the expression on the right
    b. Assign it to the variable on the left (varnameL)
    c. Print one integer numbers: the value of varnameL

Examples:

INPUT	OUTPUT

`x1 = 12`
`x2 = 3`
`x3 = x2 * x1`
`x4 = x3 - x1`
`QUIT`
	`12`
	`3`
	`36`
	`24`

`x1 = 2`
`x2 = x1 * x1`
`x3 = x2 * x1`
`QUIT`
	`2`
	`4`
	`8`

`x2 = 12`
`QUIT`
	`12`

`x1 = 12`
`x1 = 3`
	`12`
	`ERROR`

`x1 = 2`
`x2 = x3 * x3`
	`2`
	`ERROR`

`x1 = 2`
`x2 = x2 * x1`
	`2`
	`ERROR`

Hints and suggestions:
Create a “variable” class holding, for a variable, the following information:

- name – for simplicity, it can be an integer, just the part after the “x”, e.g. for x876 it’s just 876
- value – an integer (e.g. after x876 = 2, this variable would be assigned “2”).

Then, you need an efficient way to store these objects, and to do search for them. You need a “symbol table”, a storage space holding information about the symbols (including variables like x876). In our calculator, the symbol table will store the objects defining the variables (objects of the “variable” class defined above), one object per variable, with the information outlined above (in a compiler for e.g. Java, symbol table would also store type of the variable, its location in memory, etc.).

Symbol table is typically coded using a hash table. In the hash table, use the variable name (the integer part of it) stored in the object as the key for the whole object.

You are allowed to use Java implementation of hash tables.

Submissions:
Due date: **Monday, April 18th, 5pm.** Submit through Canvas: **upload a single java source code file, named cmsc401.java (your class should be named cmsc401)**. your name in a comment at the top of the file. Do not use packages in your code. The command *javac cmsc401.java* should be sufficient to compile your code, and *java cmsc401* to run it.

Note:

- You should assume there will be one test case at a time. You do not need to think about handling multiple test cases. Grading will be done using automated script. It will be handled there.
- You should assume that inputs are always as described above. E.g., there will be no line with y1 = x2 + z3 (invalid variable names), or line x1=1 (no spaces), or line x1=0.5 (not an integer).
- The only error handling needed is for the errors causing ERROR output described above.
- Do not print extra lines such as: “Please enter..”, “The output is..”, etc.