Project #2 (The URCalculator)

CSC 172 (Data Structures and Algorithms), Spring 2018,
University of Rochester
Due Date: Monday 03/19/2018 (11:59 pm)
Try to finish the project before spring recess!

Objectives

- Practice using `Stacks` (and perhaps `Maps`). We are going beyond `Lists`!
- Learn how expression parsing and evaluation (Shunting-yard algorithm) work.

The URCalculator

For this project, you need to implement a simple calculator application. The calculator should be able to perform all of the basic operations like

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)
- Assignment (=)
- Clear variables
- Show variables and their values

What To Do

You are to write a Java program `URcalculator` that does the following:

1. It keeps reading user's inputs, line by line. Each input line the user types is supposed to be one of the following forms:

   (a) Any mathematical expression or assignment.
   (b) `clear all`
   (c) `show all`
   (d) `clear varX`
   (e) `exit`

   For example, the following commands are valid:

```
> a=b=c=d=50+(2*3 )/2
> a+b
> VarE = a + { 50 + [30.52] }
> f = +++++a----b
> g = ++a + (b + c+ (+d - -VarE))
> a = a+b+c -[{(4+2)*32.9}/9]
> clear b
> show all
> clear all
> exit
```

2. If the user types `exit`, then your program just quits.
3. If the expression is **not well-formed**, it should be reported. Example: 2 + ((5+4) (Parenthesis mismatch)
4. If the expression is **invalid**, it should be reported. Example: 3 4 +
5. If any variable is not **defined** it should be reported.
6. Any attempt to **divide by zero** should be reported.

You can create as many classes as required. But you need to name the main class as `URCalculator.java`

Note:

- No GUI for this project is allowed. You may add additional features to your calculator (for example, trigonometric functions, exponential, power of a number, log, etc) for extra credit. **But, No GUI of any sort would be allowed.**
- You can create as many classes as required. But you must name the main class as `URCalculator.java`
- For our test cases, we will not use variables longer than 10 characters, but ideally you program should work with variables of any length.
- Expression may or may not contain spaces.
- You can assume the datatype for all the numeric values is **double**.
- Assignment operations should not output anything
- 'clear all' will delete all entries from the symbol tables and won't output anything
- 'clear' followed by a valid variable name will remove the variable from the symbol table. If the variable name is invalid, it should be reported. This command should not print anything.
- 'show all' will print the symbol table, i.e, all the valid symbols and their values. No invalid/cleared symbols should be displayed.
- You can assume 'all' is a reserved keyword and won't be used as a variable name.
- You do not need to handle cases like '**' or '//'.

- For a series of '+' and '-', you need to use the known logic,
  '+' and '+' is '+'
  +' and '-' is '-'
  '-' and '-' is '+'
- You can assume all the assignments will be at the beginning of the expression. We will NOT use any test cases like the following:
  a = b + c = 50
  You can ignore cases like this.
- Most importantly, your program should print the result of any mathematical expression/operations (those does not involve assignment).

Suggestions

- You should use a `Map` for storing the symbol table.
- I will explain in class how to use the `Stack` data structures in the context of this problem.

How to submit

Zip (archive) all the project source files and a README file together and save it as `Proj2.zip`. Upload the file to Blackboard.

Your README file should include any specific instruction which is useful for the project. It should also include all the features (including additional features) that you have implemented.

Make sure all your source files are properly commented so that user can browse your code without difficulty.

Grading

This is an individual project. Total points: 100

- `0 points` if the program does not compile.
- `5 points` if the `exit` command works.
- `15 points` if the program detects and reports 'not well-formed' for not-well-formed expressions
- `15 points` for identifying and reporting three cases, invalid operations, not defined variables, and attempts to divide by zero.
- `65 points` if the program works without any issue. We will test your program through a series of test cases.
- Maximum extra credit possible: 20 pts.
- Again, NO GUI of any kind would be allowed for this project, not even for extra credit.