CSE214

HOMEWORK 3 - SPRING 2025

HOMEWORK 3 - due Tuesday, March 11th no later than 6:00PM

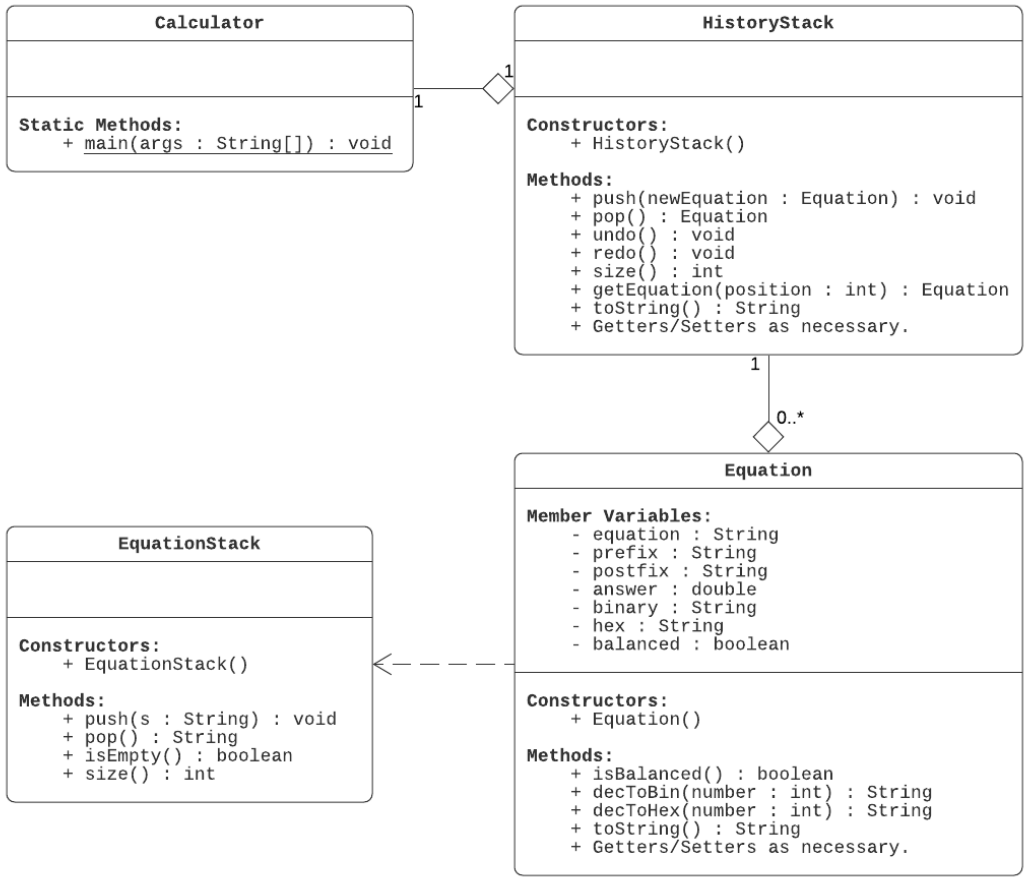
REMINDERS:

- **Submit your files only via CodeGrade on the Content page in Brightspace. Access CodeGrade by clicking "HW3 - CODEGRADE SUBMISSION LINK" for your submission.**
- **Use of a package is optional. If you wish to use it, make sure to name it "hw3" (all in lowercase). Otherwise, you will lose points.**
- **If your program doesn't run on CodeGrade or if your score on CodeGrade is less than 50%, the maximum grade you can receive for this assignment is 90.**
- **Be sure your code follows the coding style for CSE214.**
- **Make sure you read the warnings about academic dishonesty. *Remember, all work you submit for homework or exams MUST be your own work.***
- **You may use any Java API class that you wish.**
- **You may use Scanner, InputStreamReader, or any other class that you wish for keyboard input.**
- ***CAUTION: Although the Java API Stack class has push and pop methods, the Stack class is a subclass of Vector. Therefore, all of the methods of Vector are also accessible in the Stack class. However, if you use any of the inherited Vector methods in your solution, you will be penalized in this assignment, since some of the Vector methods are not supposed to apply to a Stack ADT in general. (That is, the designers of Java basically define that a Stack is a special type of Vector, but it really isn't.)***

You have recently been hired by Wolfy Tech, an up and coming software company based out of Stony Brook, NY. Wolfy Techs' main goal is to make the world a better place by using cutting edge technology in everything from refrigerators to notepads. You have been hired as a Junior Software Developer to help design their newest line of calculators. You may be thinking this will just be some ordinary calculator, but you quickly realize at the first production meeting that Wolfy Tech only makes the highest quality of products, and your calculator will be much more advanced than originally thought. Not only will the calculator have to calculate numbers, but it will also need to show these calculations in pre-fix / post-fix notation and show the answer in binary and hexadecimal, in addition to some other proprietary features. After some research and preliminary write-ups and brainstorming, you realize that the best way to go about this will be to create your own Stack class.

UML

The UML Diagram for all the classes specified below is as follows:



SPECIFICATIONS

1. Equation

Write a fully documented class named Equation. This class will store the actual equation as a String (in in-fix notation), pre- and post-fix notations as separate Strings, the answer in decimal (base 10) as a double, the answer in binary (base 2) as a String, the answer in hex (base 16) as a String, and a boolean balanced flag to denote whether the equation is balanced or not. Below are some methods you will need:

- `public Equation()`
  Constructor for Equation (you may include an additional constructor with parameters.
- `public boolean isBalanced()`
  Determines if the equation is balanced or not. This means that the equation has properly matched parentheses. Returns true if the equation is balanced, false otherwise.
- `public String decToBin(int number)`
  Converts `number` from decimal (base 10) to binary (base 2) and returns that value. If the equation is not valid (not balanced), throw an appropriate exception.
- `public String decToHex(int number)`
  Converts `number` from decimal (base 10) to hexadecimal (base 16) and returns that value. If the equation is not valid (not balanced), throw an appropriate exception.
- `public String toString()`
  Returns a String which is a neat, tabular representation of this Equation. See Sample I/O.
- **Getters/Setters for all fields as necessary.**
- **NOTE: When converting between the different bases, you must do the math yourself, and not rely on a Java conversion function.**

2. EquationStack

Write a fully-documented class named EquationStack that you will use to evaluate an Equation to get the answer. It is recommended to extend the Java Stack API when implementing this class. You can keep track of the different types of stacks using a variable, depending on whether the stack is to store operators or operands. Below are some methods you will need:

- `public EquationStack()`
  Constructor for EquationStack (you may include an additional constructor with parameters).
- `public void push(String s)`
  Pushes s (a new operator or operand) to the top of the stack.
- `public String pop()`
  Removes the operator or operand stored at the top of the stack.
- `public boolean isEmpty()`
  Returns true if the stack is empty, false otherwise.
- `public int size()`
  Returns the current size of the stack.

3. HistoryStack

Write a fully-documented class named HistoryStack that will contain the user's history of equations. This will be similar to the above EquationStack, with some extra functionality.

- `public HistoryStack()`
  Constructor for HistoryStack (you may include an additional constructor with parameters).
- `public void push(Equation newEquation)`
  Adds newEquation to the top of the stack.
- `public Equation pop()`
  Removes and returns the Equation at the top of the stack.
- `public void undo()`
  Removes the last Equation from the top of the stack, preserving it for later.
- `public void redo()`
  Replaces the last undone Equation back onto the stack. If there is no last undone Equation, throw an appropriate exception.
- `public int size()`
  Returns the current size of the stack.
- `public Equation getEquation(int position)`
  Searches through this HistoryStack and returns the Equation located at the specified position. If the position is out of range or otherwise invalid, throw an appropriate exception.
- `public String toString()`
  Returns a neatly formatted table of this HistoryStack. See Sample I/O.
- **NOTE: The HistoryStack should be set up so that the very first equation entered will be located at position 1, and for the position of each pushed equation after that, the position will increase by 1.**
- **NOTE: The table should be printed in decreasing order (the newest equation in history should be the first to be printed).**
- **HINT: Consider using another stack for undo/redo.**

4. Calculator

Write a fully-documented class named Calculator that will represent the company's new calculator. This class will contain the main method that acts as a driver for the program that the user will interact with.

- `public static void main(String[] args)`
  - **The main method that allows the user to use the calculator.**
  - **Menu:**
    - `A`**: Add a new equation.**
      Create and solves a new equation in infix notation.
    - `F`**: Change equation in history.**
      Locates an equation from the history, and will then replace, remove, or add an operator or operand to that equation based on the user's input. Then add it to the top of the history stack without replacing the original equation.
      When modifying the equation, wait until the user has completed all changes to the equation before testing it for validity and evaluating it.
    - `B`**: Print previous equation.**
      Prints the last equation entered.
    - `P`**: Print full history.**
      Prints the user's history in a neatly formatted table.

- **`U`: Undo last equation.**
  Removes the equation at the top of the history.
- **`R`: Redo last equation which was undone.**
  Replaces the last equation which was undone at the top of the history.
- **`C`: Clear history.**
  Resets the calculator to its initial state.
- **`Q`: Quit**
  Quits the program.

5. You will need classes to handle the exceptions thrown (see class specifications for exceptions classes you need). Make sure every exception is accounted for, and that the program does not end prematurely.

NOTE: You may include additional methods, variables, or classes as necessary or as you find convenient.

INPUT FORMAT

- Each menu operation is entered on its own line, and should be case insensitive (i.e., 'q' and 'Q' are the same).
- When the user is entering equations, all operands should be entered as integers, and operators may include the symbols +, -, /, *, % (modulo), and ^ (exponentiation).
- Check to make sure the position, if required, is valid. If not, print an error message and return to the main menu.
- All user input should be case insensitive.
- If the equation causes an error (such as a dividing by 0, or no numbers, or anything that would make you be unable to calculate an answer), it should be marked as not balanced, prefix and postfix should be set to N/A, and answer / binary / hex should all be set to 0.

OUTPUT FORMAT

- Each command should out the result (as shown in the Sample I/O below) after each operation is performed.
- All menu operations must be accompanied by a message indicating what operation was performed and whether or not it was successful.
- The decimal answer must be rounded to 3 decimal places.

SAMPLE INPUT/OUTPUT

```
//Comments in green, input in red, output in black.

Welcome to calculator.

[A] Add new equation
[F] Change equation from history
[B] Print previous equation
[P] Print full history
[U] Undo
[R] Redo
[C] Clear history
[Q] Quit

Select an option: A

Please enter an equation (in-fix notation): (360*5)-2
The equation is balanced and the answer is 1798.000

// Menu not shown in sample i/o

Select an option: A

Please enter an equation (in-fix notation): (5839/23 + ((68-7) % 8))
The equation is balanced and the answer is 258.870

// Menu not shown in sample i/o

Select an option: A

Please enter an equation (in-fix notation): (2+1
The equation is not balanced.

// Menu not shown in sample i/o

Select an option: B

#   Equation                        Pre-Fix                     Post-Fix                    Answer              Binary  Hexadecimal
-------------------------------------------------------------------------------------------------------------------------------
3   (2+1                            N/A                         N/A                          0.000                   0            0

// Menu not shown in sample i/o

Select an option: P

#   Equation                        Pre-Fix                     Post-Fix                    Answer              Binary  Hexadecimal
-------------------------------------------------------------------------------------------------------------------------------
3   (2+1                            N/A                         N/A                          0.000                   0            0
2   (5839/23 + ((68-7) % 8))        + / 5839 23 % - 68 7 8      5839 23 / 68 7 - 8 % +     258.870           100000011          103
1   (360*5)-2                       - * 360 5 2                 360 5 * 2 -               1798.000          11100000110          706

// Menu not shown in sample i/o
```

```
Select an option: F

Which equation would you like to change? 2

Equation at position 2: (5839/23 + ((68-7) % 8))
What would you like to do to the equation (Replace / remove / add)? Replace
What position would you like to change? 2
What would you like to replace it with? 9

Equation: (9839/23 + ((68-7) % 8))
Would you like to make any more changes? Y // Can be entered as 'Y', 'YES', 'N', or 'NO' (case-insensitive)
What would you like to do to the equation (Replace / remove / add)? remove
What position would you like to remove? 5

Equation: (983/23 + ((68-7) % 8))
Would you like to make any more changes? >Yes
What would you like to do to the equation (Replace / remove / add)? Add
What position would you like to add something? 5
What would you like to add? 1

Equation: (9831/23 + ((68-7) % 8))
Would you like to make any more changes? n
The equation is balanced and the answer is 432.435

// Menu not shown in sample i/o

Select an option: P

#   Equation                        Pre-Fix                         Post-Fix                          Answer              Binary  Hexadecimal
-------------------------------------------------------------------------------------------------------------------------------------------
4   (9831/23 + ((68-7) % 8))        + / 9831 23 % - 68 7 8          9831 23 / 68 7 - 8 % +          432.435           110110000          1B0
3   (2+1                            N/A                             N/A                               0.000                   0            0
2   (5839/23 + ((68-7) % 8))        + / 5839 23 % - 68 7 8          5839 23 / 68 7 - 8 % +          258.870           100000011          103
1   (360*5)-2                       - * 360 5 2                     360 5 * 2 -                    1798.000          11100000110          706

// Menu not shown in sample i/o

Select an option: U

Equation '(9831/23 + ((68-7) % 8))' undone.

// Menu not shown in sample i/o

Select an option: U

Equation '(2+1' undone.

// Menu not shown in sample i/o

Select an option: P

#   Equation                        Pre-Fix                         Post-Fix                          Answer              Binary  Hexadecimal
-------------------------------------------------------------------------------------------------------------------------------------------
2   (5839/23 + ((68-7) % 8))        + / 5839 23 % - 68 7 8          5839 23 / 68 7 - 8 % +          258.870           100000011          103
1   (360*5)-2                       - * 360 5 2                     360 5 * 2 -                    1798.000          11100000110          706

// Menu not shown in sample i/o

Select an option: A

Please enter an equation (in in-fix notation): (5*6)
The equation is balanced and the answer is 30.000

// Menu not shown in sample i/o

Select an option: R

Redoing equation '(2+1'.

// Menu not shown in sample i/o

Select an option: P

#   Equation                        Pre-Fix                         Post-Fix                          Answer              Binary  Hexadecimal
-------------------------------------------------------------------------------------------------------------------------------------------
4   (2+1                            N/A                             N/A                               0.000                   0            0
3   (5*6)                           * 5 6                           5 6 *                            30.000               11110           1E
2   (5839/23 + ((68-7) % 8))        + / 5839 23 % - 68 7 8          5839 23 / 68 7 - 8 % +          258.870           100000011          103
1   (360*5)-2                       - * 360 5 2                     360 5 * 2 -                    1798.000          11100000110          706


// Menu not shown in sample i/o

Select an option: A

Please enter an equation (in in-fix notation): (2^3^2)
The equation is balanced and the answer is 512.000
// Note: exponentiation is evaluated from right to left

// Menu not shown in sample i/o

Select an option: P

#   Equation                        Pre-Fix                         Post-Fix                          Answer              Binary  Hexadecimal
-------------------------------------------------------------------------------------------------------------------------------------------
5   (2^3^2)                         ^ 2 ^ 3 2                       2 3 2 ^ ^                        512.000          1000000000          200
4   (2+1                            N/A                             N/A                               0.000                   0            0
3   (5*6)                           * 5 6                           5 6 *                            30.000               11110           1E
2   (5839/23 + ((68-7) % 8))        + / 5839 23 % - 68 7 8          5839 23 / 68 7 - 8 % +          258.870           100000011          103
1   (360*5)-2                       - * 360 5 2                     360 5 * 2 -                    1798.000          11100000110          706


// Menu not shown in sample i/o
```

```
Select an option: C
Resetting calculator.
// Menu not shown in sample i/o
Select an option: R
There is nothing to redo.
// Menu not shown in sample i/o
Select an option: u
There is nothing to undo.
// Menu not shown in sample i/o
Select an option: P
#   Equation                          Pre-Fix                          Post-Fix                          Answer              Binary  Hexadecimal
-------------------------------------------------------------------------------------------------------------------------------------------------
// Menu not shown in sample i/o
Select an option: Q
Program terminating normally...
```
