Class Design and Testing

Chase Carroll, CPSC4900, Assignment 1: SimpleCalculator Java

Class Design

The SimpleCalculator class was arranged into three main methods, four utility methods, and two free variables. A main() method was used for driving the program by taking user input for one of three options: HELP (to repeat a help message), EXIT (to terminate), or an infix expression.

Main Methods	Utility Methods	Free Variables
calculate	isNumeric	postOrder
postfix	round	operators
doTheMath	prec	
	print	

Main Methods

calculate: Drives the program by calling the other two main methods and returning their result. Implemented via the interface of the same name. Having this be the controller of the class helps during initial prototyping and limits the number of methods that need called in main().

postfix: Attempts to convert infix expressions to postfix expressions. Given that evaluating an infix expression requires repeated scanning over the whole of it to preserve order of operations, postfix is a much more efficient format, given that order of operations is guaranteed if the postfix is valid. This method includes numerous checks to try to prevent illegal expressions from being accepted. The final result of the conversion or a caught error is returned to doTheMath.

doTheMath: This method evaluates postfix expressions and performs minimal error checking, in the case that postfix returned a "Invalid expression" string. This method returns a string answer to calculate.

Utility Methods

isNumeric: Determines if a given string is representative of a number or not. Needed for evaluating tokens of the postfix expressions to determine if they are operands or operators.

round: Performs rounding on the final answer returned to calculate from doTheMath. Enables for accurate rounding up to 5 decimal places without truncation.

prec: Determines precedence among operators to ensure order of operations is followed when converting from infix to postfix.

print: Method to simplify system output for quicker typing. Not necessary to have, but if your tab-completion does not work on System.out.println(), this method is useful.

Free Variables

postOrder: ArrayList of Strings to provide persistence of the infix-to-postifx expression outside of the postfix method. Breaks up tokens of the expression so that they do not have to be parsed from a string.

operators: String array of accepted operator symbols. Used for reference by the postfix method for sifting out unsupported non-numeric characters.

Testing

Testing was performed with JUnit 4 via Eclipse IDE's built-in support for it. Testing methodology was based around calling the calculate method with increasingly complex and lengthy expressions and then checking the returned value against a precomputed result. Some expressions were created by hand and others were obtained from the internet via searching "infix expressions for junit tests" until a satisfactory amount was acquired. Finally, testing to check for proper "Invalid expression" responses was performed using a handful of erroneous infix expressions.