CS 251 Intermediate Programming
Lab 5: Postfix Calculator

Brooke Chenoweth

Fall 2018

Problem Description

In this assignment, you will implement a postfix calculator. This program allows a user to enter operands and operators at the command line and calculates the result as a postfix calculator would.

You will get some experience with nested classes and with Java collections.

What do you need to do?

I have provided you with the following interfaces and classes:

- `Stack` – generic interface defining stack methods
- `StackTest` – testing class for your `StackOfDoubles` implementation.
- `Operator` – interface for operator objects
- `CalcTest` – testing class that reads tokens from standard input and evaluates them with a postfix calculator

1. Write a `StackOfDoubles` class that implements `Stack<Double>`. You may *not* use the `java.util.Stack` class for this. However, you may use a Java Collection class in your implementation. (I suggest looking at `LinkedList` for this. Let the library code do the heavy lifting here. Don't reinvent the wheel.)
2. Write a `PostfixCalculator` class.
   - You should have two private member variables in this class
     - A stack of `double` operands
     - A map of strings to operator objects. (something like `private Map<String, Operator> operatorMap;`)

- The constructor should fill the operator map with assocations of symbols to operator objects as described in the table below. You will have to create multiple implementations of the operator interface in order to do this.

  The `Operator` implementations should be nested inside the `PostfixCalculator` class. It is up to you if you make them static nested classes, non-static inner classes, local classes, or anonymous classes, but they must all be inside `PostfixCalculator` somehow.
- The `storeOperand` method takes a `double` and pushes it onto the operand stack. It does not return anything.
- The `evaluateOperator` method takes an operator string, looks up the corresponding operator object in the operator map, pops the appropriate number of operands (as given by the `numArgs` method) and places them into a list, evaluates the operator with the operands in the list, and pushes the result onto the operand stack. It does not return anything, but operators may have side effects, such as printing the argument to standard output.

3. Test your code.

Operators

Your calculator should recognize the following operators. Note that there can be more than one symbol for a given operation. (For example, `1 2 +` would have the same result as `1 2 add`)

Name	Symbol	# Arguments	Result
addition	`+` `add`	2	Sum of arguments A B + → A+B
subtraction	`-` `sub`	2	Difference of arguments A B - → A-B
multiplication	`*` `mult`	2	Product of arguments
division	`/` `div`	2	Quotient of arguments

print	`=`
	`print`
		1	Print argument to standard out
			and return argument

Turning in your assignment

Once you are done with your assignment, use UNM Learn to turn in `StackOfDoubles.java` and `PostfixCalculator.java`. You should not have changed any of the classes I provided you, so do not submit them.