CAS CS 112 – Spring 2012, Programming Assignment 2

due at 10:00 pm on Thursday, February 16

Problem 2, Calculator

(70 points) In this assignment you will write a programmable calculator. Your implementation will consist of a client plus four classes: **Calculator, Bag, Variable, and ArrayBasedStack**.

Calculator

(20 Points) Your client will use the Calculator API, which consists of a constructor plus two methods that you must implement: `int evaluate (String)` and `void print()`. Strings passed to evaluate() consist of one-character variables, one-digit integer values, arithmetic operators (+ , - , * , /), and the assignment operator (=). Valid strings are assignments of the form `<var> = <expr>`, where `<var>` is a one-character variable name and `<expr>` is a valid postfix expression (examples below). The print() method simply prints all values of variables.

```
Calculator c = new Calculator();
c.evaluate("x=5");
c.evaluate("y=x7+");
c.evaluate("z=x7+4x*+");
c.evaluate("w=xy+");
c.print();
//This code should print "x: 5, y: 12, z: 32, w: 17"
```

Collection

(15 Points) To implement your Calculator, you will need a Collection data structure to store variables and their values. In this assignment, you will implement a Bag class using generics and doing so with a linked-list implementation (feel free to work from the code in the textbook). Your Calculator should implement a Bag of Variables, where Variable is a class you should create that just has two member variables (no methods): a name, which is a char, and a value, which is an int. Bags and Variables should be defined in separate classes named **Bag** and **Variable**.

Iterator

(10 points) Your **Bag** should implement the **Iterable** interface. You should use your Iterator in your Calculator methods when looking up and printing the values of variables.

Stack

(15 Points) You will need a stack of integers to evaluate postfix expressions. The stack can be fixed size, i.e. it is fine to work from the ArrayBased implementation in the book. You should only need one stack to perform each evaluation. The stack should also be its own class named **ArrayBasedStack**.

Handling bad input

(10 points) If your calculator encounters a malformed postfix expression or an undefined variable, it must throw a **malformedPostfixException** or **undefinedVariableException**, respectively. We will show you how to define your own exceptions by extending the Exception class in lab.

Submissions

Please WebSubmit all files necessary for compilation, including the files named above and any additional files that you used. You may submit a test file including a main method if you wish.