Computer Science 501
Data Structures & Algorithms
The College of Saint Rose
Fall 2015

Lab 8: P.S.: It's Just a Stack

Due: 6:00 PM, Friday, November 6, 2015

In addition to finishing up last week's leftover tasks (the sorting algorithms empirical study), you will be writing one more program to practice working with a stack.

You may work alone or in a group of 2 or 3 for this assignment.

Getting Set Up

To get your BlueJ environment set up for this week's lab assignment, start BlueJ and choose "New Project" from the "Project" menu. Navigate to your folder for this course and choose the name "Lab8" (no spaces) for the project.

Create a document where you will record your answers to the lecture assignment and lab questions. If you use plain text, call it "`lab8.txt`". If it's a Word document, you can call it whatever you'd like, but when you submit, be sure you convert it to a PDF document "`lab8.pdf`" before you submit it.

Lecture Assignment Questions

We will usually discuss these questions at the start of class on the lab due date, so no credit can be earned for late submissions of lecture assignment questions.

Note: Don't forget to submit the two lecture assignment questions from last week (9.10 and 9.12). These will still be graded as part of last week's lab.

? LA Question 1:

(6 points) In the last lab, you were asked to iterate through all subsets of a set (its power set). This is an important operation for many *brute-force algorithms* like the one you are tackling in that lab.

A related problem is to generate all *permutations* of a set, also often an important mechanism for solving a problem by brute force. How many permutations are there of a list of n items? Describe an algorithm (either a pencil and paper approach or a psuedocode algorithm) that would generate all permutations.

As an example, the permutations of the set 123 would be: 123, 132, 213, 231, 312, 321. (Hint: I find this ordering – the *lexicographic* ordering to be the most natural way to think about generating these. Think first about how you would generate the permutations of 1-element, 2-element, 3-element, and 4-element sets, then see if you can generalize.)

? LA Question 2:
Bailey Problem 9.4, p. 212. (3 points) [Be sure to do the one in the "Problems" section at the bottom of the page, not the one in the "Self Check Problems" section at the top.]

? LA Question 3:
Bailey Problem 9.16, p. 213. (3 points)

? LA Question 4:
Bailey Problem 10.8, p. 246. (2 points)

Postscript

Your programming assignment this week is described in Section 10.5 of Bailey. You will learn about the PostScript language, and gain experience using stacks and making use of an existing partial implementation.

Getting Started

First, carefully read the lab description in the text and this handout. You are certain to have questions, so be sure to leave plenty of time to discuss this assignment with me.

Before you begin, copy the starter files from the the shared area under `labs/postscript`. Familiarize yourself with these three classes and make sure you understand how they work.

Next, develop a sketch of the `Interpreter` class. This is where you will process the tokens being delivered to your interpreter by the provided `Iterator` class.

Notes and Guidelines

- Please name the file with your interpreter and `main` method `Interpreter.java`.
- Make use of the functionality of the classes you are given to start. Be careful not to spend time developing code that replicates functionality that is already there!
- Make your `main` method very short. All it should do is create an `Interpreter` object and tell it to start parsing the PostScript program presented at the command line. Create a method `interpret` that takes a single parameter of type `java.util.Iterator<Token>` (which will be an instance of the provided `Reader` class) and processes the PostScript tokens returned by that `Iterator<Token>`.
- A "debugging mode" for your program may prove useful to facilitate debugging and testing. In my version, if I specify the word "debug" as a command-line parameter, the program enters a debugging mode and prints out informative messages about what it is doing as it processes the PostScript tokens.

- Develop your `interpret` method incrementally. Get the simple opertations like pushing a token encountered on the input, `pop`, and `pstack` working, then move on to the arithmetic operators, and finally the definition and usage of symbols.
- Your program should throw exceptions when it encounters invalid input, but these should contain meaningful error messages. You can use `Assert.condition()` and `Assert.fail()` for this.

Bonus Opportunity

For two bonus points, you can implement procedures as described in thought question 3. If you design everything else properly, this should be almost a trivial extension.

For another two bonus points, you can implement the `if` operator as described in thought question 4.

Related Questions

Question 1:
Answer thought question 1 from Bailey p. 249. (1 point)

Question 2:
Answer thought question 2 from Bailey p. 249. (2 points)

Question 3:
Answer thought question 5 from Bailey p. 250. Note: while you don't have to do questions 3 and 4, you need to read and understand them to be able to do 5. (2 points)

Submitting

Before 6:00 PM, Friday, November 6, 2015, submit your lab for grading. There are two things you need to do to complete the submission: (i) Copy your file with the answers to the lecture assignment and lab questions into your project directory. Be sure to use the correct file name. If you prepared your answers in Word, export to a PDF file and submit that. (ii) Email a copy of your lab (a `.7z` or `.zip` file containing your project directory) to *terescoj@strose.edu*. Please use a meaningful subject line such as "Joe Student Lab8 Submission".

Grading

This assignment is worth 45 points, which are distributed as follows:

Feature	Value	Score
LA Question 1 (permutations)	6	
LA Question 2 (9.4)	3	
LA Question 3 (9.16)	3	
LA Question 4 (10.8)	2	
`Interpreter` program design and style	4	
`Interpreter` program documentation	6	
`Interpreter` correctness	16	
Question 1 (Thought Question 1)	1	
Question 2 (Thought Question 2)	2	
Question 3 (Thought Question 5)	2	
Bonus opportunity 1: procedure support	(2)	
Bonus opportunity 2: if support	(2)	
Total	45	

The program design grade will be based on the design choices you make in the implementation of the `Interpreter` class. The program style grade will be based on code formatting and approriate use of Java naming conventions. The program documentation grade is, of course, based on the comments you provide. The program correctness grade is based on how well your program meets the functionality requirements.