Programming and Data Structures
Active Learning Activity 6: Using Data Structures

Activity Objectives

At the end of this activity, students should be able to:

1. Use the generic data structure ***`Stack`*** from the Java Collection Framework to evaluate arithmetic expressions.

2. Use the generic data structure ***`PriorityQueue`*** from the Java Collection Framework to simulate a simple printing queue with priority.

Part 1: Using ***`java.util.Stack`***

1. Create an instance of the generic class ***`Stack`*** for the type ***`Integer`*** and name it ***`postfixStack`***.
2. Prompt the user to enter a postfix expression with operands and operators separated by a space. Extract the tokens from the input postfix expression using the method ***`split()`*** from class ***`String`***.
3. Evaluate the postfix expression using the following algorithm:
    a. Consider one token from the input expression.
    b. If the token is an operand, push it in the stack.
    c. If the token is an operator, pop two values from the stack, perform the operation, and push the result back into the stack.
    d. Repeat from **a.** until all the tokens have been processed.
    e. Pop the result of the postfix expression from the stack.
    f. If the stack is empty, display the popped value as the result, otherwise display "*`postfix expression malformed.`*"
4. Test your program for the following postfix expressions.

```
12 25 5 1 / / * 8 7 + -  (correct)
8 2 + 3 * 16 4 /          (malformed)
70 14 4 5 15 3 / * - - / 6 + (correct)
```

Part 2: Using ***`java.util.PriotiyQueue`***

Use a priority queue to simulate how the printer processes printing requests. Requests for printing are called jobs and are identified with three attributes: the id of the job, the priority group to which the job belongs (0 to 5 with group 0 having the highest priority), and the size of the job in bytes. Create a class ***`Job`*** with the job attributes, getters, setters, and the ***`toString`*** method. Define a natural ordering for the objects of type ***`Job`*** such that jobs are ordered based on the job's group. A sample sequence of requests is provided in the file `jobs.txt`.

1. Create an empty priority queue, read the list of jobs from the file `jobs.txt`, and add the jobs to the priority queue using the method ***`offer`***. ***`offer`*** will insert the jobs in the priority queue using their natural ordering.

2. Process the jobs in the priority queue one by one as a printer would and determine the time required to complete printing each of the requests if the printer speed is 10,000 bytes/second.

3. Display the list of jobs in the order processed by the printer and their completion time. Display the total time to print the list of jobs.

Note: The size of the printing jobs should be displayed in bytes, KB, MB, or GB depending on the size of the job. The completion time should be expressed in days, hours, minutes, and seconds. See the provided sample output for the format.

Submit the files ***`Job.java`*** and ***`Test.java`*** on Github.

A sample run of the program is shown below.

```
Evaluating postfix expressions
-------------------------------------------------------------
Enter a postfix expression:
8 2 + 3 * 16 4 /
Malformed expression.
Do you want to evaluate another postfix expression? (yes/no):
yes
Enter a postfix expression:
70 14 4 5 15 3 / * - - / 6 +
Result = 8
Do you want to evaluate another postfix expression? (yes/no):
no


Simulating printing jobs
-------------------------------------------------------------
Job ID         Job Group      Size           Completion time
33333          0              225.0KB        00:00:00:23
11001          0              7.3GB          08:09:23:20
66666          0              750.0MB        00:20:50:00
99999          1              949.0KB        00:00:01:34
55555          1              52.0KB         00:00:00:05
22222          1              15.0KB         00:00:00:02
77777          2              82.8KB         00:00:00:08
11111          2              10.0MB         00:00:16:40
44444          3              20.0MB         00:00:33:20
88888          3              25.0KB         00:00:00:03


Total Printing time: 09:07:05:34
```