Homework 1 - Building a *Binary Search Tree* & more

COP3503 — Assigned: January 22, 2020

Michael McAlpin, Instructor — Due: February 9, 2020

1 Objective

Build a Java program that will support the creation of a *Binary Search Tree*, hereinafter referred to as a *BST*. This program will support reading a command file that supports insertion, deletion, searching, printing, and subtree children and depth counts. All output will be to either `STDOUT` or `STDERR`.

2 Requirements

1. Read the input file formatted as follows. The input file will contain at least one command per line, either insert, delete, search, print, or quit. These are defined in detail below. For example, one of the input files, named **in5.txt** contains the following:

   ```
   i 9
   i 24
   i 3
   i 4
   i 11
   p
   q
   ```

2. The specific commands are **i** for insert, **d** for delete, **s** for search, **p** for print, and **q** for quit.

   (a) Insert
   The insert command uses the single character **i** as the command token. The command token will be followed by a single space then an integer.
   (This command's success can be verified by using the *print* command.)

   (b) Delete
   The delete command uses the single character **d** as the command token. The command token will be followed by a single space, then an integer.
   In the event that the *integer* cannot be found, the program will issue an error message to **STDOUT** and recover gracefully to continue to accept commands from the input file.

   ```
   command-> d 100: integer 100 NOT found
   ```

   (This command's success can be verified by using the *print* command.)

(c) Search

The search command uses the single character **s** as the command token. The command token will be followed by a single space, then an integer.

In the event that the *integer* cannot be located, the program will issue an error message to **STDOUT** and recover gracefully to continue to accept commands from the input file.

```
command-> s 101: integer 101 NOT found
```

(d) Print

The print command uses the single character **p** as the command token. This command will invoke the *print* function which will output the data in the tree *inorder*.
This command is critical for verification of all the commands specified above.

(e) Quit

The quit command uses the single character **q** as the command token. In the event the quit command is invoked, the program exits. There is no requirement for data persistence.

2.1 Functions

While there are no specific design requirements (with one exception), it might be a meaningful suggestion to consider breaking this problem into several small classes. For example, a *BST* class and a *Node* class would seem to be the minimal set of classes.

2.1.1 Required Function(s)

- **complexityIndicator**

  Prints to **STDERR** the following:

  - NID
  - A difficulty rating of how difficult this assignment was on a scale of 1.0 (easy-peasy) through 5.0 (knuckle busting degree of difficulty).
  - Duration, in hours, of the time you spent on this assignment.
  - Delimit each field with a **semicolon** as shown below.
  - Sample output:

    ```
    ff210377@eustis:~/COP3503$ ff210377;3.5;18.5
    ```

- **countChildren** which will count *all* nodes on the left branch of the BST, and then the right branch.

- **getDepth** which will provide the depth of the right and left branches of the BST.

3 Testing

Make sure to test your code on Eustis **even if it works perfectly on your machine** . If your code does not compile on Eustis you will receive a 0 for the assignment. There will be eight (8) input files and seven (7) output files provided for testing your code, they are respectively shown in Table 1 and in Table 2.

Filename	Description
input01.txt	Insert 7 integers with 2 searches and a delete the prints the tree.
in5.txt	Five integers inserted with no duplicates. Prints the tree.
in5del2.txt	Five integers added followed by two deletes. One will be a delete of a non-existent integer. Prints the tree.
in5del1srch1.txt	Five names added, one valid delete, followed by a valid search, then an invalid search. Prints the tree.
in10.txt	10 integers inserted with no duplicates. Prints the tree.
in100.txt	100 integers inserted. Prints the tree
in100m5000.txt	100 random integers (all modulo 5,000) inserted with random deletes.
in10k-m5000.txt	10,000 integers (all modulo 5,000) inserted with random deletes.
in1m-m5000.txt	1,000,000 integers (all modulo 5,000) inserted with random deletes.
mega5.txt	5,000,000 random integers with random deletes. (***For entertainment value, as it takes a long time to complete***.)

Table 1: Input files

The expected output for these test cases will also be provided as defined in Table 2. To compare your output to the expected output you will first need to redirect *STDOUT* to a text file. Run your code with the following command (substitute the actual names of the input and output file appropriately):

```
java Hw01 inputFileName > inputFileNameSt.txt
```

The run the following command (substitute the actual name of the expected *input file name* concatenated with either **St** for the student generated code or with **Valid** for the validation file:

`diff output.txt` *`inputFileName`*`St.txt` *inputFileName*Valid.txt

If there are any differences the relevant lines will be displayed (note that even a single extra space will cause a difference to be detected). If nothing is displayed, then congratulations - the outputs match!

If your code crashes for a particular test case, you will not get credit for that case.

4 Submission - via WebCourses

The Java source file(s). Make sure that the *main* program is in Hw01.java.

Use reasonable and customary naming conventions for any classes you may create for this assignment.

5 Sample output

```
ff210377@eustis:~/COP3503$ java Hw01 in10.txt
in10.txt contains:
i 888
i 77
i 90
i 990
i 120
i 450
i 7900
i 7000
i 500
i 65
p
q
 65 77 90 120 450 500 888 990 7000 7900
left children:          6
left depth:             5
right children:         3
right depth:            3
ff210377;3.5;18.5
ff210377@eustis:~/COP3503$ java Hw01 >5in-myOutput.txt
ff210377;3.5;18.5
ff210377@eustis:~/COP3503$ diff 5in-myOutput.txt 5in-expectedOutput.txt
mi113345@eustis:~/COP3503$
```

Note The **ff210377;3.5;18.5** output shown above is the output from the *complexityIndicator* function to **STDERR**.

Command	Output filenames
java Hw01 in5.txt	in5Valid.txt
java Hw01 in5del1srch1.txt	in5del1srch1Valid.txt
java Hw01 in5del2.txt	in5del2Valid.txt
java Hw01 in100.txt	in100Valid.txt
java Hw01 in100m5000.txt	in100m5000Valid.txt
java Hw01 in10.txt	in10Valid.txt
java Hw01 in10k-m5000.txt	in10k-m5000Valid.txt
java Hw01 in1m-m1000.txt	in1m-m1000Valid.txt

Table 2: Commands with input files and corresponding output files.

6 Grading

Grading will be based on the following rubric:

Percentage	Description
-100	Cannot compile on *Eustis*.
-100	Cannot read input files.
- 25	Cannot insert an integer into the BST correctly.
- 25	Cannot search for an integer in the BST correctly. This includes a search for a non-existent integer.
- 25	Cannot print the contents of the BST correctly.
- 25	Cannot delete an matching entry in the BST correctly. This includes the error case of correctly handling an attempted delete of a non-existent integer.
- 25	Cannot print the child count & depth of the left and right branches of the **root**.
- 10	Output does not match *expectedOutput.txt* exactly.

Table 3: Grading Rubric