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

COP3503 Assigned: February 5, 2021
Michael McAlpin, Instructor Due: February 28, 2021

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.)

   - For **insertion** or **realigning** operations, less than goes left, equal to or greater than goes right.

   (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 - NOT deleted
   ```

(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) Missing numeric parameter for the **Insert**, **Delete**, or **Search** command.

- Prints the following to **STDOUT**

  ```
  command-> i missing numeric parameter
  ```

  Where the command, in this case **i**, reflects the command from the input file, respectively an **i**, **d**, or **s**.

(f) 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.

2.1.2 Code Requirements

- Header - the following comment block should be at the beginning of the source file.

```
/*==========================================================================
 |   Assignment:  HW 01 - Building and managing a BST
 |
 |       Author:  Your name here
 |     Language:  Java
 |
 |   To Compile:  javac Hw01.java
 |
 |   To Execute:  java Hw01 filename
 |                     where filename is in the current directory and contains
 |                           commands to insert, delete, print.
 |
 |        Class:  COP3503 - CS II Spring 2021
 |   Instructor:  McAlpin
 |     Due Date:  per assignment
 |
 +==========================================================================*/
```

- The following *Academic Integrity Statement* should be at the end of the source file.

```
/*==========================================================================
 |     I [your name] ([your NID]) affirm that this program is
 | entirely my own work and that I have neither developed my code together with
 | any another person, nor copied any code from any other person, nor permitted
 | my code to be copied  or otherwise used by any other person, nor have I
 | copied, modified, or otherwise used programs created by others. I acknowledge
 | that any violation of the above terms will be treated as academic dishonesty.
 +==========================================================================*/
```

- Java Library Support
  - The goal of this assignment is to build resources for BST. Therefore it is not acceptable to use the many resources built in to Java to streamline *Tree* management. For example it is not acceptable to use the `java.util.TreeSet` or similar libraries.

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 eight (8) output files provided for testing your code, they are respectively shown in Table 1 and in Table 2.

Filename	Description
badCmd.txt	2 integers inserted. Search for valid integer, then searches with no specified integer, likewise for another insert, and a delete. Prints the tree.
in1.txt	Inserts one integer and 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.
in100m50K.txt	100 random commands of insert or delete for random integers (all modulo 50,000).
in10K-m50M.txt	10,000 random commands of insert or delete for random integers (all modulo 5,000,000).

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*** `>` ***inputFileName*****`St.txt`**

Then 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 -w` ***inputFileName*****`St.txt`** *inputFileName***Valid**.txt

If there are significant differences the relevant lines will be displayed (note that the `diff` command parameter, `-w`, is used to ignore whitespaces). If nothing is displayed, then con-

gratulations - the outputs match!

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

3.1 Implementation notes & testing

3.1.1 Implementation

- The input file is read into an array list, then once read in, is processed from the array list. *It is acceptable to use the* `ArrayList` *class.*
- Once the input file is read into the array, it is then printed, and, finally, processed.
- The decision for left and right insertions is simple.
  - If the input integer is **less than** the examined node in the tree, it will be inserted to the **left**.
  - If the input integer is **NOT less than** the examined node in the tree, it will be inserted to the **right**.

3.1.2 Testing

- The `bash` script, `testHw01.sh`, when *unzipped* will not have *execute permission* for the file. Execute the following command in the terminal window: `chmod +x *.sh`. *Make sure the command is executed in the directory containing the script file.*
- Executing the script can be accomplished by entering the following command:

  ```
  bash testHw01.sh
  ```

- Make sure the **SINGLE** source code file is named `Hw01.java`.

4 Submission - via WebCourses

The Java source file named `Hw01.java`. Make sure that the *main* program is in `Hw01.java` and that it contains all the classes needed in that **single** file.

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	Valid baseline output filenames
java Hw01 badCmd.txt	badCmdValid.txt
java Hw01 in1.txt	in1Valid.txt
java Hw01 in5.txt	in5Valid.txt
java Hw01 in5del2.txt	in5del2Valid.txt
java Hw01 in5del1srch1.txt	in5del1srch1Valid.txt
java Hw01 in10.txt	in10Valid.txt
java Hw01 in100m50K.txt	in100m50KValid.txt
java Hw01 in10K-m50M.txt	in10K-m50MValid.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* given the bounds of ignoring *whitespaces*. (`diff -w inputFileName baseFileName`)

Table 3: Grading Rubric