COMP 282 Summer 2019 **Project 2**

Due Saturday, June 29th 2019 23:55

Requirements: To complete this project you will write and submit 1 file: `Project2.zip`

Project2.zip is a zip-file archive containing **all of your source code, as .java files**. You do not need to include .class files in your Project2.zip. You must also include a plain-text file inside the zip-file named `Readme.txt`

Each of these files will be created as an ASCII file (i.e. a plain text document with a .java or .txt extension). You may create your source code and `Readme.txt` with any editor or IDE but you must ensure that they are plain text files. In other words, they should not contain anything except ASCII characters.

`Readme.txt` must contain your answers to the questions at the bottom of this assignment. See section named "QUESTIONS FOR README.TXT"

NOTE: you may use any standard Java libraries for this assignment, but you must implement the AVL tree as a full binary search tree with rotations and balance factor, as discussed in lecture. You may not use any Java library, standard or otherwise, nor code from the internet for this AVL tree implementation!

IMPLEMENTATION

Create an AVL tree as a re-balancing binary search tree of type ***int***. **You may not perform lazy deletion**, which means the tree must potentially re-balance upon each insertion and deletion. The tree should have…

- insert(Node nd)
  - inserts, potentially re-balancing
- delete(Node nd)
  - deletes, potentially re-balancing
- inOrderTraversal()
  - Print **only the integer values** in sorted order
  - Example: you can print the traversal of a tree with a root (value 42) and 2 children (values 11 and 50) as:
    - **`11 42 50`**
- lookup(int value)
  - returns a Node (reference to the node object in the tree) or null (not found in tree)
- print()
  - Print the tree nodes in this form…
  - You can represent each node as (*address, value, left-child-address, right-child-address*). The root should be printed first. "Null pointers" are represented by special address -1.
  - Print as a preOrderTraversal
  - Example: you can represent a tree with a root (value 42) and 2 children (values 11 and 50) as:
    - **`(1003402, 42, 1004404, 2008101) (1004404, 11, -1, -1) (2008101, 50, -1, -1)`**

Suggestion: you may want to add parent pointers and balance-factor to your tree nodes, and maybe temporarily print these as well to aide your debugging. Additionally you may use a debugger.

QUESTIONS FOR README.TXT

1) What, if anything, does not work in this code?
2) What, if anything, is incomplete in this code?
3) Show that this code works across 5 different integer inputs by using the inOrderTraversal() and print() methods.
    a. Note: you should have a main(…) method that prompts a user for input, or reads input from command-line. This method is up to you, but it must work to get input for the AVL tree.
    b. You must show successful inOrderTraversal() and print() for ***five different int trees containing at least 15 nodes each***, and these trees must be automatically balanced as well as automatically printed in sorted order.

SUBMISSION

Prior to the deadline upload your file `Project2.zip` to Canvas. Do not submit any other documents!

GRADING CRITERIA

10%: Submission instructions followed to the letter (1 zip file named Project2.zip, containing appropriate Readme.txt and all required .java files for compilation)

20%: Readme.txt answers questions

30%: All source code compiles and executes as specified

40%: Source code looks proper and matches specification

A Word On Cheating: This project is an individual project. You can discuss this project with other students. You can explain what needs to be done and give suggestions on how to do it. You cannot share source code. If two projects are submitted which show significant similarity in source code then both students will receive an F in the course. Note a person who gives his code to another student also fails the class (you are facilitating the dishonest actions of another).