Project 3 – A Binary Tree Implementation

Due: November 27 , 2022 **Total point value: 100**

Goals

The goal of this project is to gain experience implementing and working with a Binary Tree. Further, students will receive some experience working with Generics.

Project Description

You have been asked by your instructor, who is not particularly clever with this description, to write a program that allows you to store data in a binary tree structure. Effectively, you are to write a simple database, allowing users to add nodes to your binary tree, remove nodes from your binary tree, and search for nodes in the binary tree. The particular data stored in the tree is up to you, as this project requires that the binary tree be generic.

A few suggestions for amusing data items:

- Dodos
- Books
- The Severed Heads of Your Foes, Who Challenged Your Claims and Titles
- Music Albums

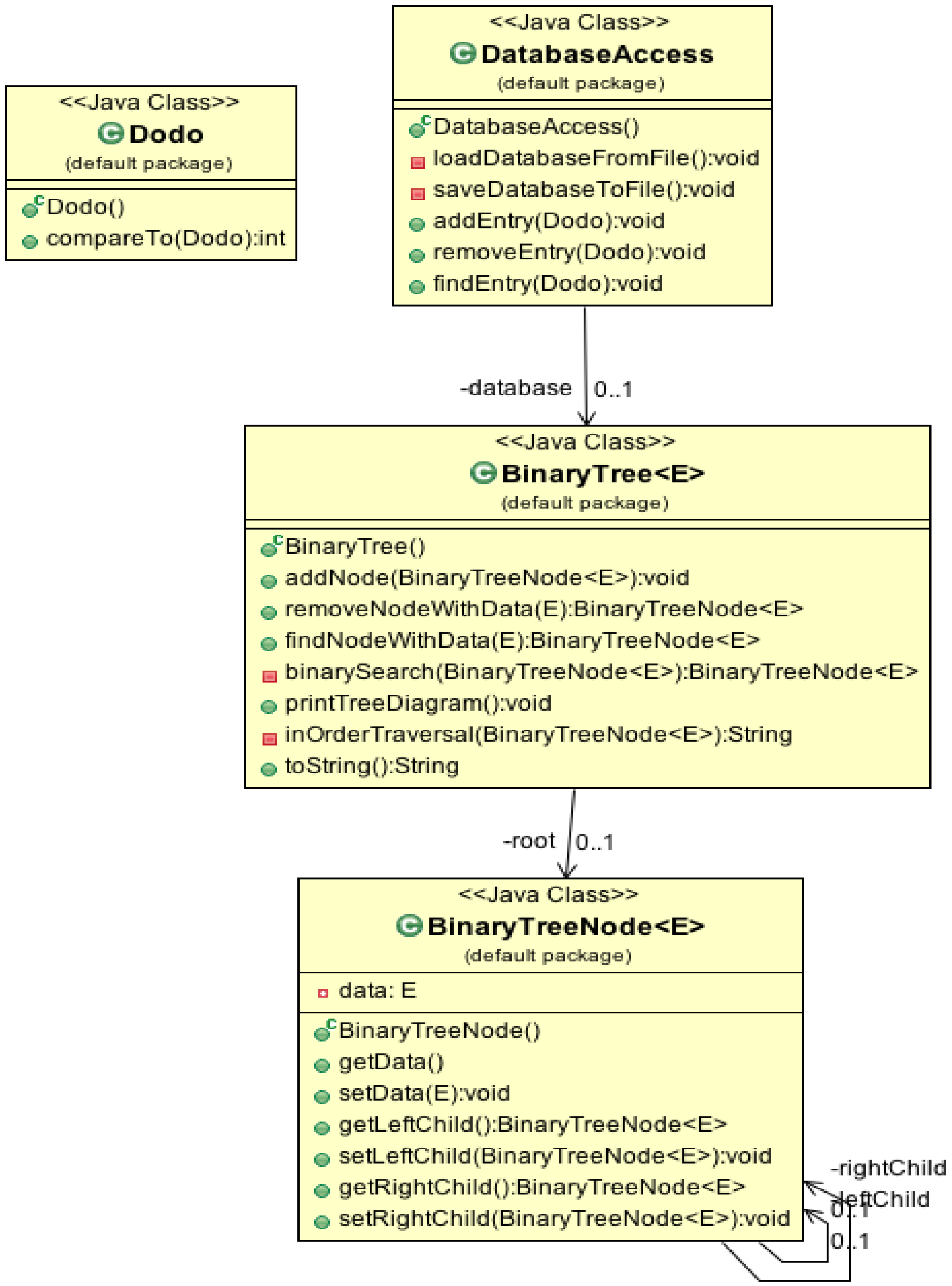
Implementation Requirements

Located on the next page is a simple UML for the basic class set up. Below are a few added details.

- The tree, tree node, and your data class should implement Serializable, to be savable and loadable
- Dodo
    - This is what I used for my data :D. Left off are the fields, as I did not feel like specifying them for the UML. You can use any fields and data you like
- DatabaseAccess
    - This is the "main", as well as the way to interact with the tree
    - Specified are the add, remove, and find functions
    - Also specified are saving and loading the database
    - Not specified is how you print the tree and interact
        - You can use a GUI or a command line interface
        - Nicer GUIs may get bonus points
    - Should save and load the same database tree for EACH RUN of the program
        - Use the same file name each time

- BinaryTree is the actual BinaryTree for the database

    - findNodeWithData uses the binarySearch function, which starts at the root as the argument
    - You can implement add and remove as you wish
    - The class should support two kinds of printing – printTreeDiagram which prints the diagram top down, and toString, which uses inOrderTraversal to represent the tree as a sorted list
- BinaryTreeNode represents an individual node in the tree
- Final notes
    - The generic types MUST be bound to extend Comparable
    - Your data item, such as Dodo, must implement Comparable



Coding Style and GUI Style

You will be graded on following proper OOP principles and basic coding style. No bad variable names etc.

Bonus points will be awarded for using a nice GUI over a command line interface

AVL Rotations and Balancing

Extra credit will be given for implementing AVL and Balancing. No help will be given in this, as it is extra credit :P

Comments

Each class and method should have a Javadoc (/** */) style comment, explaining what the class/method does. @param should be used to describe each argument, while @return should be used to describe return types. I suggest letting Eclipse's autocomplete generate those tags on at least one method so that you get an example of the format to base the rest on!

Submit: Submit a zip of your .java Files on Brightspace. YOUR CODE MUST COMPILE TO BE GRADED.

Evaluation:

Requirements	**Pts**	**Comment**
Correctness	**80/80**	
Code Compiles	10/10	
Data item follows requirements	10/10	
BinaryTreeNode set up well	10/10	
BinaryTree	**35/35**	
Supports add and removal	10/10	
Supports binarySearch	10/10	
Proper tree printing	15/15	
DatabaseAccess	**15/15**	
Support operations with tree	10/10	
Properly saves/loads database(the tree) each time	5/5	
Design	**10/10**	
Documentation and Style	**10/10**	
Bonus: GUI is extra nice	**5/0**	**Grader discretion**
Bonus: AVL implementation	**5/0**	**Grader discretion vs completeness of implementation**