Project PL4

Due Date: December 2

Purpose

You will write a program to store various trees using Java and its OOP capabilities.

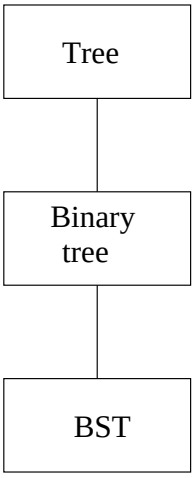
This is an individual project.

Problem

In Algorithms, you learned about all sorts of trees:

- generic tree
- binary tree
- binary search tree (BST)
- 2-3 tree
- B-tree
- heap
- etc.

In this project, you will create various trees and display them in a nicely formatted way. The project should handle generic, binary, and binary search trees. Using Java's OOP capabilities, create an object hierarchy as shown below:



Each node in the tree will store a simple single character.

Input

All of the input will be done interactively by the user via the keyboard. The program should prompt for:

1. the max number of items (characters) to store.
2. what sort of tree the user wishes to create; the input for this is up to you.

3. if the chosen type is a generic tree, then the number of branches from a node will be input next. This will then be constant for that tree.

4. the data; the user will input one character per line, with a 0 (zero) indicating the end of input. Items should be added to generic trees and binary trees from the top-down and left-to-right. In a BST, alphabetic ordering should be preserved.

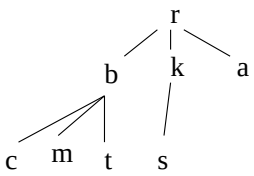
Once the tree is created, the program should repeatedly give the following options:

- **'s':** search for item x in the tree and return "found" or "not found". This method is also useful for deleting an item, below.
- **'r':** remove (delete) item x from the tree or state "not found."
- **'a':** add item x into the tree. For a generic or binary tree, the item should be inserted into the topmost and leftmost available node. In other words, fill in the tree from top-to-bottom, left-to-right. (This is similar to a heap.)
- **'t':** traverse the tree. This means display all of the nodes. In the case of a binary tree or BST, this should do an inorder traversal. In the case of a generic tree, it should do a preorder traversal, where the order of the branches followed should be from left-to-right.
- **'d':** display the tree.
- **'q':** quit the program.

Obviously, if an option requires an item x to be input, the program should prompt for that as well.

Output

As an example, if the tree were created with the characters r, b, k, a, c, m, t, s and each node had 3 branches, then the resulting tree would be:



For the traversals, the program should display up to 10 nodes on a line, with a space between each character. For the tree above, the preorder traversal would be:

```
r b c m t k s a
```

To display a tree, the program should display each level of the tree (that is, all of the nodes at the same height) on a separate line, with one space between each character. Empty nodes on a level should display a 0 (zero). For the above tree, the display would look as follows:

```
r
b k a
c m t s 0 0 0 0 0
```

Stop displaying data when the maximum height is reached.

Specifics

- The program must be written in Java.
- The program should follow the design shown above; that is, you should have a super class and two derived classes.
- A main objective of OOP is to promote code reuse. Therefore, something that is done in the super class (such as drawing a tree) should not be redone in a derived class.
- The tree should be stored in an array. You will need additional data to support the different kinds of trees.
- Encapsulate data as much as possible. Do not make methods or data members public unless they absolutely need to be.
- Write a comment at the top of your program as in PL3. Also write a comment at the start of each method.

Notes

- Submit your source code via email in a zip or tar file named with your last name and PL4 as in `gousiePL4.zip` or `gousiePL4.tar`. You will have at least three Java files. Hand in a hard copy version of your program in the envelope on my door by 5:00 PM on Friday, 12/3.

The answer is either `m` *or something else.*
– Eva Ma, one of my grad school professors