## Overview

You will write a Java application to build and search a B-Tree. This will include:  
Node, RootNode, LeafNode classes, plus a main class of some sort

You will need to make test cases for your tree. You can build the tree in your main class by calling the constructors. The graders will create their own tree(s) by modifying your code and test those.

## Data Structures

You will need to build a Node object to be the base class for the following two node types:

RootNode - holds a start of the range, end of the range and some number of Nodes (can be RootNodes or LeafNodes).

LeafNode - holds any number of integer values.

There should be a single RootNode that is the "top" of the tree. Its range should encompass the range of the whole tree.

## Searching

If the current node is a RootNode and the number that we are looking for is between the start and end of the range, follow the nodes that are descendents of this RootNode. If the current node is a LeafNode, check each value of the leaf node to see if it matches the value that we are searching for.

## Rules:

1) No iterators, loops or Java functions that iterate for you. All iteration must happen by recursion.

2) No global or static variables. The class that does the searching should not have any members.

3) Please make constructors:

LeafNode(Collection<int> values)

RootNode(int min, int max, Collection<Node> nodes)

Remember to work independently. This is not a large (lines of code) assignment. It is an assignment that is designed to make you think in a different way.

***MAKE SURE TO TEST YOUR CODE!*** Make a B-Tree and search for numbers that are in it and numbers that are not in it.

Note – this is not quite a typical B-Tree; the assignment is a slightly simplified version of an actual B-Tree.

Example B-Tree diagram:

A close up of a map

Description generated with high confidence

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rubric | Poor | OK | Good | Great |
| Comments | None/Excessive (0) | “What” not “Why”, few (5) | Some “what” comments or missing some (7) | Anything not obvious has reasoning (10) |
| Variable/Function naming | Single letters everywhere (0) | Lots of abbreviations (5) | Full words most of the time (8) | Full words, descriptive (10) |
| Structure | Globals everywhere, indentation doesn’t match braces {}, no helper functions (0) | Any 2 of:  Too many globals  Indentation wrong  Missing helper functions (5) | Any 1 of:  Too many globals  Indentation wrong  Missing helper functions (13) | Few/no globals, indentation correct, helper functions(20) |
| Recursion | Doesn’t exist (0) | Functions exist, not called (7) | Most recursion exists (14) | Recursion Completely Correct (20) |
| Inheritance | Doesn’t exist (0) | Node exists (4) | Node exists, Root and Leaf inherit (7) | Polymorphic behavior present (10) |
| Constructors | Don’t exist/Empty (0) | One constructor (5) |  | Both Constructors (10) |
| Searching | Doesn’t work at all (0) | Some searches work (7) | Minor search errors (14) | All searches perfect (20) |