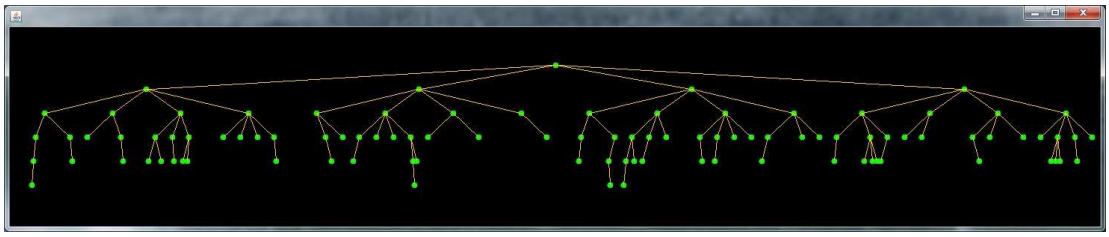
Assignment #6: Random Quadtree

In this assignment you will learn to:

- deal with a new data structure, a tree
- process data in this tree in an iterative, and a recursive way
- complete a task that is described in a way that assumes you have your 1068/2168 knowledge well together
- connect your own data structure to a given visualizer, by implementing an interface

Let me throw you in cold water: we did not talk about trees in class yet, but you will have to program a tree data structure, even quite a special one, a Quadtree. A Quadtree is a linked data structure of nodes; every node is linked to max. 4 child-nodes, see image below.



The top node is called the "root". In the example, it has 4 nodes as children (could be less, but in this example it's 4), which are again roots of smaller quadtrees. These, again, have max. 4 nodes as children, which are again roots of smaller children, which, again, have max. 4 nodes as children, which are again roots of smaller children and so on (do you feel some recursion here?).

Important for a tree (any tree) is, that there are no cycles, i.e. each node, except for the root, has exactly one parent, i.e. there is only a single node that contains a link to it; the root has no parent. Usually, trees are implemented such they only link downwards, i.e. each node contains references to the children, not to the parent.

Your task:

- Create a data structure that can represent a quadtree, i.e. create the classes QNode and QuadTree. A QNode should contain a double value and an array of QNode[4], which can contain the references to the (max.) 4 children. No reference to the parent is needed.
- In class QuadTree, create a method "insertAtRandomPosition(double v)". This method should, from the root on, create a random position index (between 0 and 3) POS and insert a new node (containing the value v) at POS, if that position is empty. If the position is not empty, it should go to the child at POS, create a new random position, check and insert if possible etc. THIS MUST BE PROGRAMMED ITERATIVELY! DO NOT USE RECURSION HERE.
- In class QuadTree, write a RECURSIVE method "sum()", that sums up all values in this tree.
- Test your structure with some code in a third class, class Main: instantiate a QuadTree qt, and fill it with values 1,2,3,4,...,100 using the insertAtRandomPosition(value) method. Then call qt.sum(). You know it should return 5050 (that's the sum from 1...100 = 101*50 = 5050).

- Connect your project to a visualizer, which I programmed. To display your tree, the visualizer takes as input an object of type "QuadTreeNode". QuadTreeNode is an interface in the same package as the visualizer. The interface declares a single method: public QuadTreeNode[ ] getChildren(). Your QNode class must implement this interface. The method "getChildren()" must return the array of children in your node. If you then instantiate the visualizer (i.e. an object of type "QuadTreeViz") using the constructor "QuadTreeVisualizer(QuadTreeNode root)" with "root" being the root node of your quadtree, your quadtree will automatically be displayed.
- The visualizer comes as a jar-file, QTvisualizer.jar. The jar-file contains
    - the package qtvisualizer, which in turn contains
        - the class "QuadTreeViz", which is the visualizer
        - the interface "QuadTreeNode", which is the interface that your QNode class has to implement

More details in class; however, this assignment description is more or less on a level that you can expect as a task description when you are working in industry (except there you would have only 1 hour to finish this :-)). Luckily you are still at Temple, and can ask the TA and me. But before you ask, try to work through it, and try to understand how all these components have to go together. It is not forbidden to google quadtrees.

Good luck!

You can find the QTvisualizer package on the class website (where you found this task description).