Assignment #7: Know Your Binary Search Tree!

This time you will program a game that is related to binary search trees (BST). It involves creation of a BST, visualization, and some user interaction.
The game in short:

- create a sequence of numbers from 1...n
- create a random permutation of that sequence
- create a BST from the permutated sequence
- visualize the structure of the BST (without showing the node-values)

This is where the interactive part starts:

- the user has to fill in the numbers 1...n into the tree at the correct place (there's a unique place for every number, which depends on the tree structure).
- subsequently, show the numbers 1..n to the user, who has to click on the respective node in the BST.
- If the solution was correct, fill in the number to that node, continue with the next number.
- If it was wrong, the user lost. Show the solution tree, and start the game again.

This assignment involves work with data structures, graphics, interaction, and game logic and therewith has a certain complexity. By now you should be able to break the task down into its parts. I will give you some help here with details about the graphics, and the interaction.

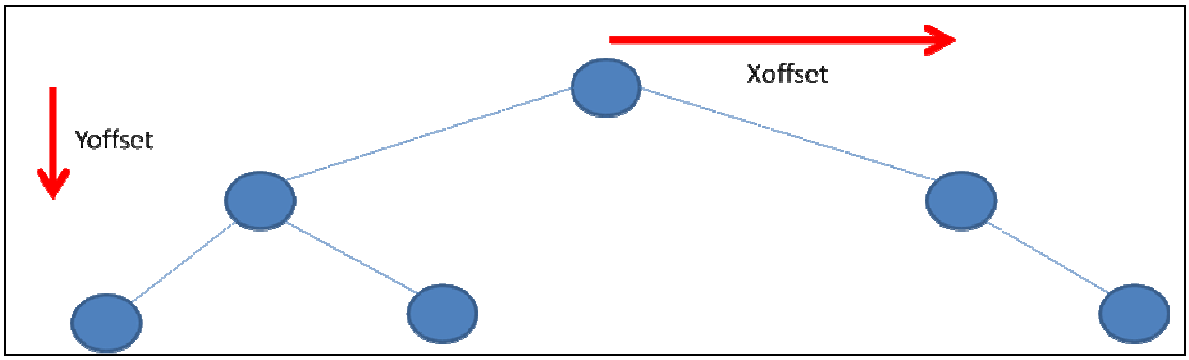
Graphics: How to Visualize the BST
Visualizing the BST involves tree traversal, yet instead of printing the value of each node, you draw the node. Here is where object oriented programming strikes: if the nodes know where and how to visualize themselves, you can decouple the traversal routine from the graphic representation: "traverse()" is a method in your BST class, while "visualize()" is a method in Node, which draws a single Node.
How do nodes know where they should be located on the screen? That is relatively simple. The node location on the screen depends on the position in the tree where the node lives. You need to know the depth of the node in the tree, and the position relative to its parent. Let's see how that works.

I would strongly suggest to have separate methods for creating the tree, and for assigning the node-positions. We always want to separate logic tasks from graphic related tasks!

Let's assume we already created the tree. In order to assign screen coordinates to each node, just traverse the tree recursively, passing down parent position, as well as an X

and Y offset to the parent. Given an initial Xoffset and Yoffset (see figure), the left child would be placed at (parentX - offsetX, parentY + offsetY). The offsetX changes depending on the depth: going one level deeper, change offsetX to offsetX/2.
Again:

- create the BST (method: BST.createFromNumberSequence(int []A)
- assign the node-positions on the existing tree (method BST.assignNodeCoordinates())
- Visualize the tree. (method: BST.visualize()). Important: this visualization is a simple tree traversal, the actual visualization has to be a method in the Node class (method Node.visualize()). This is very important! The nodes know how to visualize themselves.



Interaction:

The game part of this assignment involves interaction with the tree visualization. The player is shown a number, and must click the respective node. Hence we need a method that

- reacts to a mouse click
- checks if the click is inside of a node, and return the value of the node

SimpleGUI has mouse click methods built in, that return (x,y) of the screen position where you clicked. You only need a method "int BST.retrieveNodeValue(int x, int y)", which returns the value of a Node, given the (mouse) x,y coordinates. This method, once again, just traverses the BST, and checks for every single Node if the (x,y) coordinate lies inside of its respective representation space on the screen.
You only have to write yet another traversal, this time calling a method (boolean Node.isInside(x,y)) in the Node class. The Nodes know their positions (we assigned them before), and their size (just define a width and length), so it's easy to check if a certain x,y, coordinate is inside the box.
Btw., you are programming your own button GUI element here, congratulations.

The rest is up to you. More details in the lab!

Summary:

Minimum requirements:

Classes:

- Main
- BST
- Node

Methods:

- BST:
    - createFromNumberSequence(int []A)
    - add(int value)
    - assignNodeCoordinates()
    - visualize()
- Node:
    - visualize(...)
    - isInside(int x, int y)

Hint: the Node.visualize() method should check if the node appears

- empty --- those are the nodes which still can be clicked
- filled --- i.e. showing the correctly guessed value

For this, define a filed in Node that contains the status of the Node (empty/filled).