Binary Search Tree

Pretty Print

In this assignment you are to construct and perform transversals on a binary search tree. Your program will read from a text file (the input will be only *double*s) and output to standard out.

A binary search tree has the property that for each node (not a leaf) the *double* value stored at the node is greater than or equal to the value of its left child and less than the value of the right child.

For a given input sequence the tree is unique. Below are examples:

Input: 1, 2, 3
Binary Search Tree:

```
1
 \
   2
    \
      3
```

Input: 2, 1, 3
Binary Search Tree:

```
    2
  /   \
1       3
```

Input 3, 4, 1, 2
Binary Search Tree:

```
     3
    /  \
  1      4
   \
     2
```

The first number in the sequence is always the root of the tree, and likewise the last number is always a leaf. The following is the complete algorithm:

```
Algorithm: insertValue(double k, BinaryTree T)
    // inserts k into Binary Tree
    node parent = findParent(k, T.root())
    if k  < =  parent.value then {
            if  prarent left child = null then make parent left child with value k
            else return error
                  // or you could  findParent(k, parent left child) again
                  // but then you would use a while-loop
     }
    else {
            if  prarent right child = null then make parent right child with value k
            else return error
    }
```

The algorithm above uses the recursive algorithm below to find the parent of the new node.

```
Recursive Algorithm: findParent (double k, node v)

    // base case
    if ( k  < =  v.value and  T.leftChild(v) = null ) then  return v
    elseif ( k >  v.value and T.rightChild(v) = null ) then  return v
    else { // recursive cases
        if k  < =  v.value then  findParent (k, T.leftChild(v))
        else findParent (k, T.rightChild(v))
    }
```

The program will:

1. Print out a tree representation **on its side**
2. Print all the values of the nodes ascending order

You are to write a print routine that prints the expression tree on its left side. For each node, you should print the double value. The text for a node should be indented 4 times the depth of the node (e.g., a node at level 0, the root, would not be indented; a node at depth 3 would be indented 12 spaces).

As an example for the input: 5, 2, 6, 1, 3, 9

The output of you program is:

```
        9
    6
5
        3
    2
        1
```

In order: 1, 2, 3, 5, 6, 9

Note that the tree is on its side and the in order sequence is output after the tree representation.

The binary search tree can be implemented as a link tree or an array tree, but if you implement with an array, you cannot assume a maximum size or use built resizable classes such as *ArrayList* or *Vector*.