**Assignment 6 - Binary Trees**

**Nicholas White**

BSTNode.java

nbOfNodes()

nbOfLeaves()

reverseTree()

getAllInRange()

BinarySearchTree.java

nbOfNodes()

nbOfLeaves()

reverseTree()

getAllInRange()

**BSTNode Code**

/\*\*

\* Computes the total number of nodes in this node's subtree.

\* @return the total number of nodes in this node's subtree,

\* 1 if this node is a leaf

\*/

public int nbOfNodes() {

if (this == null){

return 0;

} else{

if (this.leftChild != null && this.rightChild != null){

return 1 + this.leftChild.nbOfNodes() + this.rightChild.nbOfNodes();

}

else if(this.leftChild != null){

return 1+ this.leftChild.nbOfNodes();

}

else if(this.rightChild != null) {

return 1+ this.rightChild.nbOfNodes();

}else{

return 1;

}

}

}

/\*\*

\* Computes the total number of leaves in this node's subtree.

\* @return the total number of leaves in this node's subtree

\*/

public int nbOfLeaves() {

if (this.leftChild == null && this.rightChild == null){

return 1;

}

else if(this.leftChild != null && this.rightChild != null){

return this.leftChild.nbOfLeaves() + this.rightChild.nbOfLeaves();

}

else if (this.leftChild == null && this.rightChild != null){

return this.rightChild.nbOfLeaves();

}

else if (this.leftChild != null && this.rightChild == null){

return this.leftChild.nbOfLeaves();

}

else{

return 0;

}

}

public void reverseTree() {

BSTNode tmp = this.leftChild;

this.leftChild = this.rightChild;

this.rightChild = tmp;

if(this.leftChild != null){

this.leftChild.reverseTree();

}

if(this.rightChild != null) {

this.rightChild.reverseTree();

}

}

public void getAllInRange(E min, E max, ArrayList<E> l) {

if (this == null) {

return;

}

if (min.compareTo(this.getValue()) < 0 && this.leftChild != null) {

this.leftChild.getAllInRange(min,max,l);

}

if (min.compareTo(this.getValue()) <= 0 && max.compareTo(this.getValue()) >= 0){

for (int i=0; i<counter;i++){

l.add(this.getValue());

}

}

if (max.compareTo(this.getValue()) > 0 && this.rightChild != null) {

this.rightChild.getAllInRange(min, max, l);

}

}

**BinarySearchTree Code**

@Override

public int nbOfNodes() {

return root.nbOfNodes();

}

@Override

public int nbOfLeaves() {

return root.nbOfLeaves();

}

@Override

public void reverseTree() {

root.reverseTree();

}

@Override

public ArrayList<E> getAllInRange(E min, E max) {

ArrayList<E> list = new ArrayList<>();

root.getAllInRange(min,max,list);

return list;

}