**BST**

How to make class that ease our work to make BST easily

It will be similar to BT

BTNode{

int data;

BTNode left;

BTNode right;

int size;

Public BTNode(){

}

Functions could be

* Traversals
  + In, pre, post, level order traversing
* Searching
* Min
* Max

}

BSTNode{

Private BTNode root;

int size;

Public BSTNode(){

}

Functions

* Insert
* Delete
* Search
* print

}

Function Search

Boolean search(int data)  
{

Return search1(root,data);

}

As we don’t want to give access of root to outside world

Private Boolean search1(root,int data)  
{

If(root.data==data)

Return true;

Else if(root.data<data)

Return Search(root.right,data);

Else

Return Search(root.left,data);

}

­­­­-function insert

Void insert(int data)

{

This.root=Insert1(root,data);

}

Private BTNode insert1(root,int data)

{

If(root==null)

{

Create node

Return node;

}

Else if(data<root.data){

Root.left=Insert1(root.left,data);

}

Else if(data>root.data){

Root.right =Insert1(root.right, data);

}

Return root;

}

-Function Delete

Here we learn how to make our own bst class

public class BinarySearchTree {

// Complete this class

private BinaryTreeNode<Integer> root;

public BinarySearchTree(){

root=null;

}

public boolean search(int data)

{

return search(root,data);

}

private boolean search(BinaryTreeNode<Integer> root,int data)

{

if(root==null)

return false;

if(root.data==data)

{

return true;

}

else if(data<root.data)

{

return search(root.left,data);

}

else if(data>root.data)

{

return search(root.right,data);

}

else

return false;

}

public void insertData(int data)

{

this.root=insert(root,data);

}

private BinaryTreeNode<Integer> insert(BinaryTreeNode<Integer> root,int data)

{

if(root==null)

{

return new BinaryTreeNode<Integer>(data);

}

else if(data<root.data)

{

root.left=insert(root.left,data);

}

else

{

root.right=insert(root.right,data);

}

return root;

}

public void deleteData(int data){

this.root=delete(root,data);

}

private BinaryTreeNode<Integer> delete(BinaryTreeNode<Integer> root,int data){

if(root==null)

{

return null;

}

if(data<root.data)

{

root.left=delete(root.left,data);

}

else if(data>root.data)

{

root.right=delete(root.right,data);

}

else{

if(root.left!=null && root.right==null)

{

return root.left;

}

else if(root.left==null)

{

return root.right;

}

else{

int data1 = findSuccessor(root.right);

root.data=data1;

root.right=delete(root.right,data1);

}

}

return root;

}

private int findSuccessor(BinaryTreeNode<Integer> root){

BinaryTreeNode<Integer> temp=root;

while(temp.left!=null)

{

temp=temp.left;

}

return temp.data;

}

public void printTree(){

print(root);

}

public void print(BinaryTreeNode<Integer> root){

if(root==null)

return ;

System.out.print(root.data+":");

if(root.left!=null)

{

System.out.print("L:"+root.left.data+",");

}

if(root.right!=null)

{

System.out.print("R:"+root.right.data);

}

System.out.println();

print(root.left);

print(root.right);

return ;

}

}