BINARY SEARCH TREES

Code for building and printing BST:

```
/*BST
for searching a key  if key >  root it will present on right side else left side
*/
public class BuiltBST{
    static class node{
            int data;
            node left;
            node right;
            public node(int data){
                this.data=data;
            }
    }
        public static node inserrt(node root, int val){
            if(root==null){
                root= new node(val);
                return root;
            }

            if(root.data>val){
                root.left=inserrt(root.left, val);
            }else{
                root.right=inserrt(root.right, val);
            }
            return root;
        }

        public static void inorder(node root){
            if(root==null){
           return;
        }
        inorder(root.left);
        System.out.print(root.data+" ");
        inorder(root.right);

        }
    public static void main(String[] args) {
        int values[]= {5,1,3,4,2,7};
        node root=null;
        for(int i=0;i<values.length;i++){
            root=inserrt(root, values[i]);
        }
        inorder(root);

    }
}
Output:
1,2,3,4,5,7
```

Code for searching a key in BST:

```
public class BSTsearch {
    static class Node{
        int data;
        Node left;
        Node right;
        Node(int data){
            this.data= data;
            this.left=null;
            this.right=null;
        }
    }

    public static Node insert(Node root, int arrval){
        if(root==null){
            root=new Node(arrval);
            return root;
        }
        if(root.data > arrval){
            root.left=insert(root.left, arrval);
        }else{
            root.right= insert(root.right, arrval);

        }

        return root;

    }
    public static boolean isfound(Node root,int key){
        if(root==null){
            return false;
        }
        if(root.data==key){
            return true;
        }
        if(root.data > key){
            return isfound(root.left, key);
        }else{
           return  isfound(root.right, key);
        }

    }
    public static void main(String args[]){
        int arr[]={5,1,3,4,2,7};

        Node root=null;
    for(int i=0;i<arr.length;i++){
        root= insert(root, arr[i]);
    }

       if(isfound(root,6)){
        System.out.println("found");
       }
       else{
        System.out.println( "not found");
       }
    }

}
Output:
Not found
```

Print in range:

```
public class printinRange {

        static class Node{
            int data;
            Node left;
            Node right;
            Node(int data){
                this.data= data;
                this.left=null;
                this.right=null;
            }
        }
```

```java
public static Node insert(Node root, int arrval){
    if(root==null){
        root=new Node(arrval);
        return root;
    }
    if(root.data > arrval){
        root.left=insert(root.left, arrval);
    }else{
        root.right= insert(root.right, arrval);

    }

    return root;

}
public static void printinRange(Node root,int k1,int k2){
    if(root==null){
        return;
    }
    if(root.data>=k1 &&root.data<=k2){
        printinRange(root.left, k1, k2);
        System.out.print(root.data+" ");
        printinRange(root.right, k1, k2);
    }else if(root.data<k1){
        printinRange(root.left, k1, k2);
    }else{
        printinRange(root.right, k1, k2);
    }
}
public static void inorder(Node root){
    if(root==null){
   return;
}
inorder(root.left);
System.out.print(root.data+" ");
inorder(root.right);

}

public static void main(String args[]){
    int arr[]={8,5,3,1,4,6,10,11,14};
    Node root=null;
for(int i=0;i<arr.length;i++){
    root= insert(root, arr[i]);
```

```
        }
        // inorder(root);
        printinRange(root, 5, 12);


}}output:
5 6 8 10 11
```

VALID BINARY SEARCH TREE:

```
public class validBST {

        static class Node{
            int data;
            Node left;
            Node right;
            Node(int data){
                this.data= data;
                this.left=null;
                this.right=null;
            }
        }

        public static Node insert(Node root, int arrval){
            if(root==null){
                root=new Node(arrval);
                return root;
            }
            if(root.data > arrval){
                root.left=insert(root.left, arrval);
            }else{
                root.right= insert(root.right, arrval);

            }

            return root;

        }

        public static void inorder(Node root){
            if(root==null){
           return;
        }
        inorder(root.left);
        System.out.print(root.data+" ");
        inorder(root.right);

        }

    public static boolean isvalid(Node root, Node max, Node min)  {
        if(root==null){
            return true;
        }

        else if(min!=null&& root.data<=min.data){
            return false;
        }
        else if(max != null&&root.data>=max.data){
            return false;
        }

        return isvalid(root.right,root , max) && isvalid(root.left,min ,root);
    }
        public static void main(String args[]){
            int arr[]={1,1,1};
            Node root=null;
        for(int i=0;i<arr.length;i++){
            root= insert(root, arr[i]);
        }
       if(isvalid(root, null, null)){
        System.out.println("valid");
       }
       else{
        System.out.println("not valid");
       }




}
}
Output:
Not valid
```

Mirror of a BST:

```
public class MIrrorOfBST {
```

```java
    public static class Node{
        int data;
        Node left;
        Node right;
        public Node(int data){
            this.data= data;
            this.left= null;
            this.right= null;
        }
    }

    public static Node mirror(Node r){
        if(r== null){
            return null;
        }
       Node leftMirror= mirror(r.left);
       Node rightMirror= mirror(r.right);

       r.left= rightMirror;
       r.right= leftMirror;

       return r;
    }

    public static void preorder(Node r){
        if(r==null){
            return;
        }
        System.out.print(r.data+" ");
        preorder(r.left);
        preorder(r.right);

    }

    public static void main(String[] args) {
        Node r= new Node(8);
        r.left= new Node(5);
        r.right= new Node(10);
        r.left.left= new Node(3);
        r.right.right= new Node(11);
        r.left.right= new Node(6);

        r= mirror(r);
        preorder(r);
    }
}
Output:
8 10 11 5 6 3
```