class Node {

int data;

Node left, right;

public Node(int item) {

data = item;

left = right = null;

}

}

class BinarySearchTree {

Node root;

BinarySearchTree() {

root = null;

}

// Insert a new node in the BST

void insert(int data) {

root = insertRec(root, data);

}

// A recursive function to insert a new node with given data

Node insertRec(Node root, int data) {

if (root == null) {

root = new Node(data);

return root;

}

if (data < root.data)

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

else if (data > root.data)

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

return root;

}

// Search a node in the BST

boolean search(int data) {

return searchRec(root, data);

}

// A recursive function to search a node in BST

boolean searchRec(Node root, int data) {

if (root == null)

return false;

if (root.data == data)

return true;

if (root.data > data)

return searchRec(root.left, data);

return searchRec(root.right, data);

}

// In-order traversal of the BST

void inOrder() {

inOrderRec(root);

}

// A recursive function to do in-order traversal of BST

void inOrderRec(Node root) {

if (root != null) {

inOrderRec(root.left);

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

inOrderRec(root.right);

}

}

}

public class Main {

public static void main(String[] args) {

BinarySearchTree bst = new BinarySearchTree();

/\* Inserting nodes into BST \*/

bst.insert(50);

bst.insert(30);

bst.insert(20);

bst.insert(40);

bst.insert(70);

bst.insert(60);

bst.insert(80);

// Print in-order traversal of the BST

System.out.println("In-order traversal of the BST:");

bst.inOrder();

// Search for a node in the BST

System.out.println("\n\nSearching for 40 in the BST:");

System.out.println(bst.search(40) ? "Found!" : "Not found.");

}

}