Ahmed Shahab

CS201 Assignment 1  
September 4, 2024

import java.util.Scanner;

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 with given data

void insert(int data) {

root = insertRec(root, data);

}

// 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;

}

// Delete a node with given data

void delete(int data) {

root = deleteRec(root, data);

}

// Recursive function to delete a node with given data

Node deleteRec(Node root, int data) {

if (root == null) {

return root;

}

if (data < root.data) {

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

} else if (data > root.data) {

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

} else {

if (root.left == null) {

return root.right;

} else if (root.right == null) {

return root.left;

}

root.data = minValue(root.right);

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

}

return root;

}

// Find the minimum value in the right subtree

int minValue(Node root) {

int minVal = root.data;

while (root.left != null) {

minVal = root.left.data;

root = root.left;

}

return minVal;

}

// InOrder traversal

void inorder() {

inorderRec(root);

}

void inorderRec(Node root) {

if (root != null) {

inorderRec(root.left);

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

inorderRec(root.right);

}

}

// PreOrder traversal

void preorder() {

preorderRec(root);

}

void preorderRec(Node root) {

if (root != null) {

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

preorderRec(root.left);

preorderRec(root.right);

}

}

// PostOrder traversal

void postorder() {

postorderRec(root);

}

void postorderRec(Node root) {

if (root != null) {

postorderRec(root.left);

postorderRec(root.right);

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

}

}

}

public class Main { // Changed class name to 'Main' to match the file name

public static void main(String[] args) {

BinarySearchTree bst = new BinarySearchTree();

Scanner scanner = new Scanner(System.in);

int choice;

do {

System.out.println("\nMenu:");

System.out.println("1) Create a binary search tree");

System.out.println("2) Add a node");

System.out.println("3) Delete a node");

System.out.println("4) Print nodes by InOrder");

System.out.println("5) Print nodes by PreOrder");

System.out.println("6) Print nodes by PostOrder");

System.out.println("7) Exit program");

System.out.print("Select an option (1-7): ");

choice = scanner.nextInt();

switch (choice) {

case 1:

System.out.println("Creating binary search tree with initial data (1, 2, 3, 4, 5, 6, 7)...");

int[] initialData = {1, 2, 3, 4, 5, 6, 7};

for (int value : initialData) {

bst.insert(value);

}

System.out.println("Binary search tree created.");

break;

case 2:

System.out.print("Enter the value to add: ");

int valueToAdd = scanner.nextInt();

bst.insert(valueToAdd);

System.out.println("Node " + valueToAdd + " added to the binary search tree.");

break;

case 3:

System.out.print("Enter the value to delete: ");

int valueToDelete = scanner.nextInt();

bst.delete(valueToDelete);

System.out.println("Node " + valueToDelete + " deleted from the binary search tree.");

break;

case 4:

System.out.println("InOrder Traversal:");

bst.inorder();

System.out.println();

break;

case 5:

System.out.println("PreOrder Traversal:");

bst.preorder();

System.out.println();

break;

case 6:

System.out.println("PostOrder Traversal:");

bst.postorder();

System.out.println();

break;

case 7:

System.out.println("Exiting program.");

break;

default:

System.out.println("Invalid choice. Please select a valid option.");

}

} while (choice != 7);

scanner.close();

}

}