```
// PROGRAM TO IMPLEMENT BST IN JAVA


import java.io.*;

class node {

    int data;
    public node left;
    public node right;

    public node(int elem)
    {
        data = elem;
        left = null;
        right = null;
    }
}

public class bst
{

    node root;
    int choice, left, right, elem;
    InputStreamReader ip;
    BufferedReader br;
    int parent;

    public bst()
    {
        choice = 0;
        root = null;
        ip = new InputStreamReader(System.in);
        br = new BufferedReader(ip);
    }

    public void input_element()
    {
        System.out.println("enter the element");
        try {
            String x = br.readLine();
            elem = Integer.parseInt(x);
        } catch (IOException e) {
            System.out.println("exception in input");
        }
    }

    public void inorder(node p)
    {
        if (p != null)
        {
            inorder(p.left);
            System.out.println(p.data);
            inorder(p.right);
        }
    }

    public void preorder(node p)
    {
```

```
    if (p != null)
    {
        System.out.println(p.data);
        preorder(p.left);
        preorder(p.right);
    }
}

public void postorder(node p)
{
    if (p != null)
    {
        postorder(p.left);
        postorder(p.right);
        System.out.println(p.data);
    }
}

public node insert(node root)
{
    if (root == null)
    {
        root = new node(elem);
    }
    else if (root.data < elem)
    {
        root.right = insert(root.right);
    }
    else if (root.data > elem)
    {
        root.left = insert(root.left);
    }
    return root;
}

public node delete(node root)
{
    if (root != null)
    {
        if (root.data < elem)
        {
            root.right = delete(root.right);
        }
        else if (root.data > elem)
        {
            root.left = delete(root.left);
        }
        else if (root.left == null)
        {
            root = root.right;
        }
        else if (root.right == null)
        {
            root = root.left;
        }
        else
        {
            elem = deletemin(root.right);
            root.data = elem;
            root.right = delete(root.right);
        }
        return root;
    }
    return root;
}

public int deletemin(node root)
{
    if (root.left == null)
    {
        return root.data;
    }
    else
    {
        return deletemin(root.left);
    }
}

public void findparent(node root)
{
    if (root == null)
    {
        parent = -1;
        return;
    }
    if (root.data == elem)
    {
        return;
    }
    parent = root.data;
    if(root.data > elem)
    {
        findparent(root.left);
    }
    else if (root.data < elem)
    {
        findparent(root.right);
    }
}

public void findchildren(node root)
{
    if (root == null)
    {
        left = right = -1;
        return;
    }
    if (root.data == elem)
    {
        left = right = 0;
        if (root.left != null)
        {
            left = root.left.data;
        }
        if (root.right != null)
        {
            right = root.right.data;
        }
    }
    if (root.data > elem)
    {
        findchildren(root.left);
    }
    else if (root.data < elem)
    {
        findchildren(root.right);
    }
}

public void search()
{
    parent = 0;
    input_element();
    findparent(root);
    if (parent != -1)
    {
        if (parent == 0)
        {
            System.out.println(elem + "is present ");
        }
        else
        {
            System.out.println(elem + "is present ");
        }
    }
    else
     System.out.println(elem + "not present");
}

public void findtype()
{
    parent = 0;
    left = right = 0;
    input_element();
    findparent(root);
    if (parent != -1)
    {
        if (parent == 0)
        {
            System.out.println(elem + "is the root node");
            return;
        }
        else
        {
            if (parent > elem)
            {
                System.out.println(elem + "is present as left child of" + parent);
            }
            else
            {
                System.out.println(elem + "is present as the right child of " + parent);
            }
        }
    }
    else
     System.out.println(" value not present\n");
}
```

```
public void getchoice()
{
     System.out.println("Enter the choice\t");
    try
    {
        String x = br.readLine();
        choice = Integer.parseInt(x);
    }
    catch (IOException e)
    {
        System.out.println("Incorrect choice");
    }

    switch (choice)
    {
        case 1:
            input_element();
            root = insert(root);
            break;
        case 2:
            input_element();
            root = delete(root);
            break;
        case 3:
                    System.out.println("Inorder traversal\n");
            inorder(root);
            break;
        case 4:
                    System.out.println("Preorder traversal\n");
            preorder(root);
            break;
        case 5:
                    System.out.println("Postorder traversal\n");
            postorder(root);
            break;
        case 6:
            parent = 0;
            input_element();
            findparent(root);
            if (parent == -1)
            {
                System.out.println("Element not present\n");
            }
            else if (parent == 0)
            {
                System.out.println(elem + "is at root\n");
            }
            else
            {
                System.out.println("Parent of " + elem + " is " + parent);
            }
            break;
        case 7:
            search();
            break;
        case 8:
            findtype();
            break;
            case 9:
                System.exit(0);
                break;
            default:
                System.out.println("Wrong choice");
                break;
        }
    }

    public static void main(String args[])
    {
        bst m;
        m = new bst();
        System.out.println("Menu\n");
        System.out.println("1.Insert\n2.Delete\n3.Inorder\n4.Preorder\n5.Postorder\n6.Find
parent\n7.Find node\n8.Find type\n9.Exit\n");
        while (true)
        {
            m.getchoice();
            System.out.println("\n");
        }
    }
}
```