

Assignment 2
COSC2007: Data Structure II
Jeel Tikiwala
239659420

Exercise No:

1 - Construct a binary search tree to keep the records of the student's names. You can insert the name of students that should follow the rules of binary search tree until press character 'N'.

Algorithm/ Pseudocode:

```
// Define a class for the nodes of the BST
// Constructor to initialize the node with a name
// Define a class for the binary search tree
// A method to insert a new node with a given name into the tree
// If tree is empty, make the new node the root
// Otherwise, start from the root and traverse the tree
// Compare the name to the name of the active node.
// If the name is smaller than the current node's name, go to the left subtree
// If the name is larger than the current node's name, go to the right subtree
```

Code:

INSERT METHOD:

```
//insert method
public static Node insert(Node root, String key)
{
    if(root == null) {
        root = new Node(key);
        return root;
    }

    //if key is
    if(key.compareTo(root.data) < 0) {
        root.left = insert(root.left, key);
    }
    else {
        root.right = insert(root.right, key);
    }

    return root;
}
```

MAIN METHOD:

```

// Q1: Construct a binary search tree
    String studentNames[] = {"Jeel", "Rishit",
"Bhakti", "Shubh", "Ashwin", "Shaan", "Diya"};
    Node root = null;

    //Multiple insert through for loop
    for(int i=0; i<studentNames.length; i++) {
        root = insert(root, studentNames[i]);
    }

```

2 - Perform the search operation to find the particular student.

Algorithm/ Pseudocode:

```

// A method to search for a node with a given name in the tree
// Navigate the tree starting at the root.
// Compare the name to the name of the active node.
// If the name is smaller than the current node's name, go to the left subtree
// If the name is larger than the current node's name, go to the right subtree
// If the name is equal to the current node's name, return true

```

Code:

SEARCH METHOD:

```

//search method
public static boolean search(Node root, String
studentNames) {
    if (root == null) {
        return false;
    }

    int comparison =
studentNames.compareTo(root.data);

    if (comparison < 0) {
        // Search in the left subtree
        return search(root.left, studentNames);
    } else if (comparison > 0) {
        // Search in the right subtree
        return search(root.right, studentNames);
    } else {
        // If root is equal to studentNames
        return true;
    }
}

```

MAIN METHOD:

```
//Q2: Perform the search operation
String searchName = "Jeel";
if(search(root, searchName )) {
    System.out.println("The name " +
searchName + " is found in the BST.");
}
else {
    System.out.println("Not Found in the
BST.");
}
```

OUTPUT:

The screenshot shows the Eclipse IDE interface with the StudentNames_BST.java file open in the editor. The code implements a binary search tree (BST) for student names. It includes methods for insertion, post-order traversal, and search. The search method is annotated with a comment //Q2: Perform the search operation. The searchName variable is set to "Bhakti". The output window shows the pre-order traversal of the BST and the result of the search operation.

```
131     if(root == null) {
132         return;
133     }
134
135     postorder(root.left);
136     postorder(root.right);
137     System.out.println(root.data+ " ");
138
139 }
140
141 //main class
142 public static void main(String[] args) {
143
144     // Q1: Construct a binary search tree
145     String studentNames[] = {"Jeel", "Rishit", "Bhakti", "Shubh", "Ashwin", "Shaan", "Diya"};
146     Node root = null;
147
148     //Multiple insert through for loop
149     for(int i=0; i<studentNames.length; i++) {
150         root = insert(root, studentNames[i]);
151     }
152
153
154
155     //Q2: Perform the search operation
156     String searchName = "Bhakti";
157     if(search(root, searchName )) {
158         System.out.println("The name " + searchName + " is found in the BST.");
159     }
160     else {
161         System.out.println("Not Found in the BST.");
162     }
163
164     //Space between 2 method
165     System.out.println();
```

Console Output:

```
The Pre-order traversal of this BST is:  
Jeel  
Diya  
Ashwin  
Rishit  
  
The In-order traversal of this BST is:  
Ashwin  
Diya  
Jeel  
Rishit
```

The screenshot shows the Eclipse IDE interface with the 'StudentNames_BST.java' file open in the editor. The code implements a binary search tree (BST) for student names. It includes methods for insertion, post-order traversal, and search. The search operation looks for the name 'Jeel'. The console output shows the tree's traversal and the search result.

```

131     if(root == null) {
132         return;
133     }
134     postorder(root.left);
135     postorder(root.right);
136     System.out.println(root.data+ " ");
137 }
138 }
139 }
140 }
141 //main class
142 public static void main(String[] args) {
143     // Q1: Construct a binary search tree
144     String studentNames[] = {"Jeel", "Rishit", "Bhakti", "Shubh", "Ashwin", "Shaan", "Diya"};
145     Node root = null;
146
147     //Multiple insert through for loop
148     for(int i=0; i<studentNames.length; i++) {
149         root = insert(root, studentNames[i]);
150     }
151
152
153
154     //Q2: Perform the search operation
155     String searchName = "Jeel";
156     if(search(root, searchName)) {
157         System.out.println("The name " + searchName + " is found in the BST.");
158     } else {
159         System.out.println("Not Found in the BST.");
160     }
161
162
163
164     //Space between 2 method
165     System.out.println();

```

Console Output:

```

<terminated> StudentNames_BST [Java Application] /Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java (13-Feb-2024, 3:30:57 pm - 3:30:57 pm) [pid: 7951]
The name Jeel is found in the BST.

The Pre-order traversal of this BST is:
Jeel
Diya
Ashwin
Rishit

The In-order traversal of this BST is:
Ashwin
Diya
Jeel
Rishit

```

The student “Harshil” is not in the BST” so it returned not found.

The screenshot shows the Eclipse IDE interface with the 'StudentNames_BST.java' file open in the editor. The code is identical to the previous one, but the search name is changed to 'Harshil'. The console output shows the tree's traversal and the search result.

```

131     if(root == null) {
132         return;
133     }
134     postorder(root.left);
135     postorder(root.right);
136     System.out.println(root.data+ " ");
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }

```

Console Output:

```

<terminated> StudentNames_BST [Java Application] /Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java (13-Feb-2024, 3:31:27 pm - 3:31:27 pm) [pid: 7968]
The name Bhakti is found in the BST.

The Pre-order traversal of this BST is:
Jeel
Diya
Ashwin
Rishit

The In-order traversal of this BST is:
Ashwin
Diya
Jeel
Rishit

```

3- Delete any one leaf node student

Algorithm/ Pseudocode

```
// A method to delete a node with a given name from the tree
// Navigate the tree starting at the root.
// If the name is smaller than the current node's name, go to the left subtree
// If the name is larger than the current node's name, go to the right subtree
// Case 1: delete the node that has no children
// If left and right nodes are null, it's a leaf
```

Code

DELETE METHOD:

```
//delete a node
    public static Node delete(Node root, String
studentNames) {
        int comparison =
studentNames.compareTo(root.data);

        if(comparison < 0) {
            root.left =
delete(root.left,studentNames);
        } else if (comparison > 0) {
            // Search in the right subtree
            root.right = delete(root.right,
studentNames);
        } else {
            // we searched and reached at the node
            //Case 1: 1st NO child - leaf node :: delete
node and replace with null
            if(root.left == null && root.right == null) {
                return null;
            }

            //Case 2: 2nd One child :: delete node and
replace with child node
            if(root.left == null) {
                return root.right;
            }
            else if(root.right == null) {
                return root.left;
            }
        }
    }
```

```
//Case 3: 3rd Two child:: replace value with
inorder succesor, delete the node for inorder
succesor

    Node IS = inorderSuccesssor(root.right);
    root.data = IS.data;
    root.right = delete(root.right, IS.data);
}

return root;
}

//inorder succesor fucntion
public static Node inorderSuccesssor(Node root) {
    while(root.left != null) {
        root = root.left;
    }

    return root;
}
```

MAIN METHOD:

```
//Q3: Delete one leaf node  
root = delete(root, "Shubh");
```

OUTPUT:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows multiple projects under "COSC1046_Labs5" and "COSC2006_Assignment1".
- Code Editor:** The current file is "StudentNames_BST.java". The code implements a BST for searching and deleting names.
- Console Output:** The output window shows the pre-order traversal of the BST.

```
> COSC1046_Labs5
> COSC1046_Practice
> COSC1047 Assignment1
> COSC1047 Assignment2
> COSC1047 Assignment3
> COSC1047 Assignment4
> COSC1047 Assignment5
> COSC1047 Assignment6
> COSC1047 Final
> COSC1047 Lab1
> COSC1047 Lab10
> COSC1047 Lab2
> COSC1047 Lab3
> COSC1047 Lab4
> COSC1047 Lab5
> COSC1047 Lab6
> COSC1047 Lab7
> COSC1047 Lab8
> COSC1047 Lab9
> COSC2006 Ass1
> COSC2006 Assignment1
> COSC2006 Assignment2
> COSC2006 Assignment3
> COSC2006 Assignment4
> COSC2006 Lab1
> COSC2006 Lab10
> COSC2006 Lab2
> COSC2006 Lab3
> COSC2006 Lab4
> COSC2006 Lab5
> COSC2006 Lab6
> COSC2006 Lab7
> COSC2006 Lab8
> COSC2006 Lab9
> COSC2007 Assignment1
> COSC2007 Assignment2
> COSC2007 Lab1
> COSC2007 Lab2
> COSC2007 Lab3
> JRE System Library [OpenJDK 17.0.6 (17.0.6)]
> src
  > (default package)
    > JeetKiwala_Assignment3.java
> COSC2007_Lab4
> DSA
> Servers
> TestSF

154
155      //02: Perform the search operation
156      String searchName = "Harshil";
157      if(search(root, searchName )) {
158          System.out.println("The name " + searchName + " is found in the BST.");
159      }
160      else {
161          System.out.println("Not Found in the BST.");
162      }
163
164      //Space between 2 method
165      System.out.println();
166
167      //03: Delete one leaf node
168      root = delete(root,"Shubh");
169
170      //04: Delete one child node
171      root = delete(root,"Shaan");
172
173      //05: Delete two child node
174      root = delete(root,"Bhakti");
175
176      //06: preorder traversal
177      System.out.println("The Pre-order traversal of this BST is: ");
178      preorder(root);
179      System.out.println();
180
181      //07: In-order traversal
182      System.out.println("The In-order traversal of this BST is: ");
183      inorder(root);
184      System.out.println();
185
186      //08: Postorder traversal
187      System.out.println("The Post-order traversal of this BST is: ");
188      postorder(root);
189      System.out.println();
190
191  }
192
193
194 }

Console X
<terminated> StudentNames_BST [Java Application] /Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java (13-Feb-2024, 3:31:27 pm - 3:31:27 pm) [pid: 7968]
The Pre-order traversal of this BST is:
Jeel
Diya
Ashwin
Rishit
```

```

StudentNames_BST.java
154
155     //Q2: Perform the search operation
156     String searchName = "Harshil";
157     if(search(root, searchName)) {
158         System.out.println("The name " + searchName + " is found in the BST.");
159     }
160     else {
161         System.out.println("Not Found in the BST.");
162     }
163
164     //Space between 2 method
165     System.out.println();
166
167     //Q3: Delete one leaf node
168     root = delete(root,"Shubh");
169
170     //Q4: Delete one child node
171     root = delete(root,"Shaan");
172
173     //Q5: Delete two child node
174     root = delete(root,"Bhakti");
175
176     //Q6: preorder traversal
177     System.out.println("The Pre-order traversal of this BST is: ");
178     preorder(root);
179     System.out.println();
180
181     //Q7: In-order traversal
182     System.out.println("The In-order traversal of this BST is: ");
183     inorder(root);
184     System.out.println();
185
186     //Q8: Postorder traversal
187     System.out.println("The Post-order traversal of this BST is: ");
188     postorder(root);
189     System.out.println();
190
191 }
192
193
194 }
195

```

Console X

```

terminated: StudentNames_BST [Java Application] /Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java (13-Feb-2024, 3:31:27 pm - 3:31:27 pm) [pid: 7968]
The In-order traversal of this BST is:
Ashwin
Diya
Rishi
Jeel
Rishit

```



```

StudentNames_BST.java
154
155     //Q2: Perform the search operation
156     String searchName = "Harshil";
157     if(search(root, searchName)) {
158         System.out.println("The name " + searchName + " is found in the BST.");
159     }
160     else {
161         System.out.println("Not Found in the BST.");
162     }
163
164     //Space between 2 method
165     System.out.println();
166
167     //Q3: Delete one leaf node
168     root = delete(root,"Shubh");
169
170     //Q4: Delete one child node
171     root = delete(root,"Shaan");
172
173     //Q5: Delete two child node
174     root = delete(root,"Bhakti");
175
176     //Q6: preorder traversal
177     System.out.println("The Pre-order traversal of this BST is: ");
178     preorder(root);
179     System.out.println();
180
181     //Q7: In-order traversal
182     System.out.println("The In-order traversal of this BST is: ");
183     inorder(root);
184     System.out.println();
185
186     //Q8: Postorder traversal
187     System.out.println("The Post-order traversal of this BST is: ");
188     postorder(root);
189     System.out.println();
190
191 }
192
193
194 }
195

```

Console X

```

terminated: StudentNames_BST [Java Application] /Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java (13-Feb-2024, 3:31:27 pm - 3:31:27 pm) [pid: 7968]
The Post-order traversal of this BST is:
Ashwin
Diya
Rishi
Jeel

```

Conclusion:

4- Delete any one student with one child node

Algorithm/ Pseudocode

```

// A method to delete a node with a given name from the tree
// Navigate the tree starting at the root.

```

```
// If the name is smaller than the current node's name, go to the left subtree  
// If the name is larger than the current node's name, go to the right subtree  
// If left is null and right is null, delete the node and replace with child node
```

Code

DELETE METHOD:

```
//delete a node  
public static Node delete(Node root, String studentNames) {  
    int comparison = studentNames.compareTo(root.data);  
  
    if(comparison < 0) {  
        root.left = delete(root.left,studentNames);  
    } else if (comparison > 0) {  
        // Search in the right subtree  
        root.right = delete(root.right,studentNames);  
    } else {  
        // we searched and reached at the node  
        //Case 1: 1st NO child - leaf node :: delete node and replace with null  
        if(root.left == null && root.right == null) {  
            return null;  
        }  
  
        //Case 2: 2nd One child :: delete node and replace with child node  
        if(root.left == null) {  
            return root.right;  
        }  
        else if(root.right == null) {  
            return root.left;  
        }  
  
        //Case 3: 3rd Two child:: replace value with inorder successor, delete the node for inorder successor  
        Node IS = inorderSuccessor(root.right);  
        root.data = IS.data;  
        root.right = delete(root.right, IS.data);  
    }  
}
```

```

    }

    return root;
}

//inorder successor function
public static Node inorderSuccessor(Node root) {
    while(root.left != null) {
        root = root.left;
    }

    return root;
}

```

MAIN METHOD:

```
//Q4: Delete one child node
root = delete(root, "Shaan");
```

- 5 - Delete any one student with two child nodes.

Algorithm/ Pseudocode

```
// A method to delete a node with a given name from the tree
// Navigate the tree starting at the root.
// If the name is smaller than the current node's name, go to the left subtree
// If the name is larger than the current node's name, go to the right subtree
// replace value with inorder successor, delete the node for inorder successor
```

Code

Delete method

```
//delete a node
public static Node delete(Node root, String
studentNames) {
    int comparison =
studentNames.compareTo(root.data);

    if(comparison < 0) {
        root.left =
delete(root.left, studentNames);
    } else if (comparison > 0) {
        // Search in the right subtree
```

```

        root.right = delete(root.right,
studentNames);
    } else {
        // we searched and reached at the node
        //Case 1: 1st NO child - leaf node :: delete
node and replace with null
        if(root.left == null && root.right == null) {
            return null;
        }

        //Case 2: 2nd One child :: delete node and
replace with child node
        if(root.left == null) {
            return root.right;
        }
        else if(root.right == null) {
            return root.left;
        }

        //Case 3: 3rd Two child:: replace value with
inorder succesor, delete the node for inorder
succesor
        Node IS = inorderSuccessor(root.right);
        root.data = IS.data;
        root.right = delete(root.right, IS.data);
    }

    return root;
}

//inorder succesor fucntion
public static Node inorderSuccessor(Node root) {
    while(root.left != null) {
        root = root.left;
    }

    return root;
}

MAIN METHOD
//main class
public static void main(String[] args) {

```

```

// Q1: Construct a binary search tree
String studentNames[] = {"Jeel", "Rishit",
"Bhakti", "Shubh", "Ashwin", "Shaan", "Diya"};
Node root = null;

//Multiple insert through for loop
for(int i=0; i<studentNames.length; i++) {
    root = insert(root, studentNames[i]);
}

//Q2: Perform the search operation
String searchName = "Harshil";
if(search(root, searchName )) {
    System.out.println("The name " +
searchName + " is found in the BST.");
}
else {
    System.out.println("Not Found in the
BST.");
}

//Space between 2 method
System.out.println();

//Q3: Delete one leaf node
root = delete(root,"Shubh");

//Q4: Delete one child node
root = delete(root,"Shaan");

//Q5: Delete two child node
root = delete(root,"Bhakti");

//Q6: preorder traversal
System.out.println("The Pre-order traversal
of this BST is: ");
preorder(root);
System.out.println();

//Q7: In-order traversal

```

```

        System.out.println("The In-order traversal of
this BST is: ");
        inorder(root);
        System.out.println();

    //Q8: Postorder traversal
    System.out.println("The Post-order traversal
of this BST is: ");
    postorder(root);
    System.out.println();

}

```

OUTPUT:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java projects and files.
- StudentNames_BST.java:** The code is displayed in the editor. It includes methods for pre-order, in-order, and post-order traversals, as well as delete operations. The code is annotated with comments indicating specific traversal types (e.g., "Q3", "Q4", "Q5", "Q6", "Q7", "Q8") and node deletion logic.
- Console View:** Shows the execution results:
 - The Pre-order traversal of this BST is:
Jeev
Divya
Ashwin
Rishit
 - The In-order traversal of this BST is:
Ashwin
Diya
tee1

The screenshot shows the Eclipse IDE interface with the project 'eclipse-workspace - COSC2007_Assignment2' open. The 'StudentNames_BST.java' file is selected in the Package Explorer. The code implements a Binary Search Tree (BST) for student names. It includes methods for insertion, deletion, and traversal (Pre-order, In-order, Post-order). The console output shows the In-order traversal of the tree containing nodes 'Ashwin', 'Diya', 'Jeev', and 'Rishit'. The Post-order traversal is also shown.

```

165     System.out.println();
166
167     //Q3: Delete one leaf node
168     root = delete(root, "Shubh");
169
170     //Q4: Delete one child node
171     root = delete(root, "Shaan");
172
173     //Q5: Delete two child node
174     root = delete(root, "Bhakti");
175
176     //Q6: preorder traversal
177     System.out.println("The Pre-order traversal of this BST is: ");
178     preorder(root);
179     System.out.println();
180
181     //Q7: In-order traversal
182     System.out.println("The In-order traversal of this BST is: ");
183     inorder(root);
184     System.out.println();
185
186     //Q8: Postorder traversal
187     System.out.println("The Post-order traversal of this BST is: ");
188     postorder(root);
189     System.out.println();
190
191 }
192
193
194 }
195

```

Console Output:

```

The In-order traversal of this BST is:
Ashwin
Diya
Jeev
Rishit

The Post-order traversal of this BST is:
Ashwin
Diya
Rishit
Jeev

```

This screenshot shows the same Eclipse IDE session with the 'StudentNames_BST.java' code. The code is identical to the first one, implementing a BST for student names with methods for insertion, deletion, and traversal. The console output shows the In-order traversal of the tree containing nodes 'Ashwin', 'Diya', 'Jeev', and 'Rishit'. The Post-order traversal is also shown.

```

158     if (root == null) {
159         System.out.println("The name " + searchName + " is found in the BST.");
160     } else {
161         System.out.println("Not Found in the BST.");
162     }
163
164     //Space between 2 method
165     System.out.println();
166
167     //Q3: Delete one leaf node
168     root = delete(root, "Shubh");
169
170     //Q4: Delete one child node
171     root = delete(root, "Shaan");
172
173     //Q5: Delete two child node
174     root = delete(root, "Bhakti");
175
176     //Q6: preorder traversal
177     System.out.println("The Pre-order traversal of this BST is: ");
178     preorder(root);
179     System.out.println();
180
181     //Q7: In-order traversal
182     System.out.println("The In-order traversal of this BST is: ");
183     inorder(root);
184     System.out.println();
185
186     //Q8: Postorder traversal
187     System.out.println("The Post-order traversal of this BST is: ");
188     postorder(root);
189     System.out.println();
190
191 }
192
193
194 }
195

```

Console Output:

```

The Post-order traversal of this BST is:
Ashwin
Diya
Rishit
Jeev

```

6 - Perform the pre-order traversal

Algorithm/ Pseudocode

```

//Preorder traversal
//check root first, then check left and then right node

```

Code

Preorder Method:

```
//Preorder traversal = root, left, right
```

```

public static void preorder(Node root) {
    if(root == null) {
        return;
    }

    System.out.println(root.data+ " ");
    preorder(root.left);
    preorder(root.right);

}

```

MAIN METHOD:

```

//main class
public static void main(String[] args) {

    // Q1: Construct a binary search tree
    String studentNames[] = {"Jeel", "Rishit",
    "Bhakti", "Shubh", "Ashwin", "Shaan", "Diya"};
    Node root = null;

    //Multiple insert through for loop
    for(int i=0; i<studentNames.length; i++) {
        root = insert(root, studentNames[i]);
    }

    //Q2: Perform the search operation
    String searchName = "Harshil";
    if(search(root, searchName )) {
        System.out.println("The name " +
    searchName + " is found in the BST.");
    }
    else {
        System.out.println("Not Found in the
    BST.");
    }

    //Space between 2 method
    System.out.println();

    //Q3: Delete one leaf node
    root = delete(root,"Shubh");
}

```

```

//Q4: Delete one child node
root = delete(root,"Shaan");

//Q5: Delete two child node
root = delete(root,"Bhakti");

//Q6: preorder traversal
System.out.println("The Pre-order traversal
of this BST is: ");
preorder(root);
System.out.println();

//Q7: In-order traversal
System.out.println("The In-order traversal of
this BST is: ");
inorder(root);
System.out.println();

//Q8: Postorder traversal
System.out.println("The Post-order traversal
of this BST is: ");
postorder(root);
System.out.println();

}

}

```

OUTPUT:

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - COSC2007_Assignment2/src/StudentNames_BST.java - Eclipse IDE
- Project Explorer:** Shows various Java projects and files under the root.
- Code Editor:** Displays the `StudentNames_BST.java` file with code related to a Binary Search Tree (BST) operations. The code includes methods for insertion, deletion, and traversal (Pre-order, In-order, Post-order).
- Console View:** Shows the output of the program, which is the Pre-order traversal of the BST, listing student names: Jeel, Diya, Ashwin, Rishit.

7 - Perform the in-order traversal

Algorithm/ Pseudocode

```
//check left node first
//then root
//then right node
```

Code:

In-order traversal method:

```
//inorder traversal = Left, root, right
public static void inorder(Node root){
    if(root == null) {
        return;
    }

    inorder(root.left);
    System.out.println(root.data+ " ");
    inorder(root.right);
}
```

MAIN METHOD:

```
//main class
public static void main(String[] args) {

    // Q1: Construct a binary search tree
```

```

String studentNames[] = {"Jeel", "Rishit",
"Bhakti", "Shubh", "Ashwin", "Shaan", "Diya"};
Node root = null;

//Multiple insert through for loop
for(int i=0; i<studentNames.length; i++) {
    root = insert(root, studentNames[i]);
}

//Q2: Perform the search operation
String searchName = "Harshil";
if(search(root, searchName )) {
    System.out.println("The name " +
searchName + " is found in the BST.");
}
else {
    System.out.println("Not Found in the
BST.");
}

//Space between 2 method
System.out.println();

//Q3: Delete one leaf node
root = delete(root,"Shubh");

//Q4: Delete one child node
root = delete(root,"Shaan");

//Q5: Delete two child node
root = delete(root,"Bhakti");

//Q6: preorder traversal
System.out.println("The Pre-order traversal
of this BST is: ");
preorder(root);
System.out.println();

//Q7: In-order traversal

```

```

        System.out.println("The In-order traversal of
this BST is: ");
        inorder(root);
        System.out.println();

        //Q8: Postorder traversal
        System.out.println("The Post-order traversal
of this BST is: ");
        postorder(root);
        System.out.println();

    }

}

```

8 - Perform the post-order traversal

Algorithm/ Pseudocode

```

//check left node first
//check right node
//then at last, node

```

Code

Postorder method

```

//postorder traversal = Left, Right, root
public static void postorder(Node root) {
    if(root == null) {
        return;
    }

    postorder(root.left);
    postorder(root.right);
    System.out.println(root.data+ " ");
}

}

```

MAIN METHOD:

```

//main class
public static void main(String[] args) {

    // Q1: Construct a binary search tree
}

```

```

String studentNames[] = {"Jeel", "Rishit",
"Bhakti", "Shubh", "Ashwin", "Shaan", "Diya"};
Node root = null;

//Multiple insert through for loop
for(int i=0; i<studentNames.length; i++) {
    root = insert(root, studentNames[i]);
}

//Q2: Perform the search operation
String searchName = "Harshil";
if(search(root, searchName )) {
    System.out.println("The name " +
searchName + " is found in the BST.");
}
else {
    System.out.println("Not Found in the
BST.");
}

//Space between 2 method
System.out.println();

//Q3: Delete one leaf node
root = delete(root,"Shubh");

//Q4: Delete one child node
root = delete(root,"Shaan");

//Q5: Delete two child node
root = delete(root,"Bhakti");

//Q6: preorder traversal
System.out.println("The Pre-order traversal
of this BST is: ");
preorder(root);
System.out.println();

//Q7: In-order traversal

```

```

        System.out.println("The In-order traversal of
this BST is: ");
        inorder(root);
        System.out.println();

    //Q8: Postorder traversal
    System.out.println("The Post-order traversal
of this BST is: ");
    postorder(root);
    System.out.println();

}

```

OUTPUT:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java projects and files under the workspace.
- StudentNames_BST.java:** The active code editor window displays the Java code for a binary search tree (BST) implementation. The code includes methods for insertion, deletion, and traversal (preorder, inorder, postorder).
- Console View:** The bottom-right window shows the output of the program. It starts with the message "The Post-order traversal of this BST is:" followed by the names Ashwin, Diya, Rishit, and Jeel, which were printed by the postorder() method.

CONCLUSION:

The above code is a Java program that implements a binary search tree (BST) of student names. The code defines a Node class that represents a node in the BST, and provides methods for inserting, searching, deleting, and traversing the tree. The main method creates a BST from an array of student names, and performs some operations on it, such as searching for a name, deleting some nodes, and printing the preorder, inorder, postorder traversal of the tree. The code demonstrates the basic concepts and operations of a BST data structure.