

Apache NetBeans IDE 22

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects > binarytree_preorder > MyMain.java

Source History

```
public class MyMain {  
    /*  
     * This function visits all boxes in postorder: left, right, root  
     */  
    public static void postOrder(Node root) {  
        // based on rule: visit left child, then right child, then current node  
        if (root != null) {  
            postOrder(root.left); // first visit left side  
            postOrder(root.right); // then visit right side  
            System.out.print(root.value + " "); // finally visit this box  
        }  
    }  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        int n = sc.nextInt(); // this reads how many nodes will be given  
        Map<Integer, Node> nodes = new HashMap<>(); // stores all boxes  
        Set<Integer> children = new HashSet<>(); // to track child nodes  
  
        int firstNodeVal = -1; // will keep the very first node (main root)  
  
        // This will read all input lines and connect the boxes  
    }  
}
```

MyMain.java - Navigator

Members > MyMain > Node

Output - binarytree_preorder (run)

```
run:  
3  
1 2 3  
2 -1 -1  
3 -1 -1  
2 3 1  
BUILD SUCCESSFUL (total time: 1 second)
```

Apache NetBeans IDE 22

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects > binarytree_preorder > MyMain.java

Source History

```
public class MyMain {  
    /*  
     * This function visits all boxes in postorder: left, right, root  
     */  
    public static void postOrder(Node root) {  
        // based on rule: visit left child, then right child, then current node  
        if (root != null) {  
            postOrder(root.left); // first visit left side  
            postOrder(root.right); // then visit right side  
            System.out.print(root.value + " "); // finally visit this box  
        }  
    }  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        int n = sc.nextInt(); // this reads how many nodes will be given  
        Map<Integer, Node> nodes = new HashMap<>(); // stores all boxes  
        Set<Integer> children = new HashSet<>(); // to track child nodes  
  
        int firstNodeVal = -1; // will keep the very first node (main root)  
  
        // This will read all input lines and connect the boxes  
    }  
}
```

MyMain.java - Navigator

Members > MyMain > Node

Output - binarytree_preorder (run)

```
run:  
4  
1 2 3  
2 4 -1  
3 -1 -1  
4 -1 -1  
4 2 3 1  
BUILD SUCCESSFUL (total time: 1 second)
```

Apache NetBeans IDE 22

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects > MyMain.java - Files Services

Source History

```
public class MyMain {  
    /*  
     * This function visits all boxes in postorder: left, right, root  
     */  
    public static void postOrder(Node root) {  
        if (root != null) {  
            postOrder(root.left); // first visit left side  
            postOrder(root.right); // then visit right side  
            System.out.print(root.value + " "); // finally visit this box  
        }  
    }  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        int n = sc.nextInt(); // this reads how many nodes will be given  
        Map<Integer, Node> nodes = new HashMap<>(); // stores all boxes  
        Set<Integer> children = new HashSet<>(); // to track child nodes  
  
        int firstNodeVal = -1; // will keep the very first node (main root)  
  
        // This will read all input lines and connect the boxes  
    }  
}
```

MyMain.java - Navigator

Members

- MyMain
- Node

Output - binarystree_postorder (run)

```
run:  
1  
5 -1 -1  
5  
BUILD SUCCESSFUL (total time: 1 second)
```

Apache NetBeans IDE 22

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects > MyMain.java - Files Services

Source History

```
public class MyMain {  
    /*  
     * This function visits all boxes in postorder: left, right, root  
     */  
    public static void postOrder(Node root) {  
        if (root != null) {  
            postOrder(root.left); // based on rule: visit left child, then right child, then current node  
            postOrder(root.right); // first visit left side  
            System.out.print(root.value + " "); // then visit right side  
            System.out.print(root.value + " "); // finally visit this box  
        }  
    }  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        int n = sc.nextInt(); // this reads how many nodes will be given  
        Map<Integer, Node> nodes = new HashMap<>(); // stores all boxes  
        Set<Integer> children = new HashSet<>(); // to track child nodes  
  
        int firstNodeVal = -1; // will keep the very first node (main root)  
  
        // This will read all input lines and connect the boxes  
    }  
}
```

MyMain.java - Navigator

Members

- MyMain
- Node

Output - binarystree_postorder (run)

```
run:  
5  
1 2 3  
2 4 5  
3 -1 -1  
4 -1 -1  
5 -1 -1  
4 5 2 3 1  
BUILD SUCCESSFUL (total time: 1 second)
```

The screenshot shows the Apache NetBeans IDE 22 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The bottom status bar shows the time as 47:27 and mode as INS.

The Projects view shows a single project named "binarytree_postorder". The Source view displays the following Java code:

```
public class MyMain {  
    /*  
     * This function visits all boxes in postorder: left, right, root  
     */  
    public static void postOrder(Node root) {  
        if (root != null) {  
            postOrder(root.left); // first visit left side  
            postOrder(root.right); // then visit right side  
            System.out.print(root.value + " "); // finally visit this box  
        }  
    }  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        int n = sc.nextInt(); // this reads how many nodes will be given  
        Map<Integer, Node> nodes = new HashMap<>(); // stores all boxes  
        Set<Integer> children = new HashSet<>(); // to track child nodes  
  
        int firstNodeVal = -1; // will keep the very first node (main root)  
  
        // This will read all input lines and connect the boxes  
    }  
}
```

The Navigator view shows the members of the MyMain class: main and postOrder. The Output view shows the results of running the program with input 7, producing the output:

```
1 2 3  
2 4 5  
3 6 7  
4 -1 -1  
5 -1 -1  
6 -1 -1  
7 -1 -1  
4 5 2 6 7 3 1  
BUILD SUCCESSFUL (total time: 2 seconds)
```

This screenshot is identical to the one above, showing the same Java code, project structure, and output results. The only difference is the time displayed in the bottom status bar, which is now 47:27.

binarytree_preorder - Apache NetBeans IDE 22

```
public class MyMain {  
    /*  
     * This function visits all boxes in postorder: left, right, root  
     */  
    public static void postOrder(Node root) {  
        if (root != null) {  
            postOrder(root.left); // first visit left side  
            postOrder(root.right); // then visit right side  
            System.out.print(root.value + " "); // finally visit this box  
        }  
    }  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        int n = sc.nextInt(); // this reads how many nodes will be given  
        Map<Integer, Node> nodes = new HashMap<>(); // stores all boxes  
        Set<Integer> children = new HashSet<>(); // to track child nodes  
  
        int firstNodeVal = -1; // will keep the very first node (main root)  
  
        // This will read all input lines and connect the boxes  
    }  
}
```

Output - binarytree_preorder (run) ×

```
run:  
2  
1 -1 2  
2 -1 -1  
2 1  
BUILD SUCCESSFUL (total time: 1 second)
```

binarytree_preorder - Apache NetBeans IDE 22

```
public class MyMain {  
    /*  
     * This function visits all boxes in postorder: left, right, root  
     */  
    public static void postOrder(Node root) {  
        if (root != null) {  
            postOrder(root.left); // based on rule: visit left child, then right child, then current node  
            postOrder(root.right); // first visit left side  
            postOrder(root.right); // then visit right side  
            System.out.print(root.value + " "); // finally visit this box  
        }  
    }  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        int n = sc.nextInt(); // this reads how many nodes will be given  
        Map<Integer, Node> nodes = new HashMap<>(); // stores all boxes  
        Set<Integer> children = new HashSet<>(); // to track child nodes  
  
        int firstNodeVal = -1; // will keep the very first node (main root)  
  
        // This will read all input lines and connect the boxes  
    }  
}
```

Output - binarytree_preorder (run) ×

```
run:  
6  
5 2 3  
2 4 -1  
3 -1 -1  
4 -1 -1  
5 -1 -1  
1 -1 6  
4 2 3 6 1 5  
BUILD SUCCESSFUL (total time: 2 seconds)
```

The screenshot shows the Apache NetBeans IDE 22 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The left sidebar has a Projects tab, Files tab, and Services tab, with the Projects tab currently selected, showing a project named 'binarytree_preorder' containing Source Packages, Test Packages, Libraries, and Test Libraries. The main workspace displays the 'MyMain.java' source code. The code implements a post-order traversal of a binary tree. It starts by reading the number of nodes from standard input. Then, it initializes a map to store nodes and a set to track child nodes. The main loop reads node values and connects them based on their index. Finally, it prints the post-order traversal of the tree. The Navigator pane shows the members of the MyMain class, including the main method and the postOrder helper method. The Output pane shows the run results, which include the input values and the resulting post-order traversal output.

```
public class MyMain {  
    /*  
     * This function visits all boxes in postorder: left, right, root  
     */  
    public static void postOrder(Node root) {  
        // based on rule: visit left child, then right child, then current node  
        if (root != null) {  
            postOrder(root.left); // first visit left side  
            postOrder(root.right); // then visit right side  
            System.out.print(root.value + " "); // finally visit this box  
        }  
    }  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        int n = sc.nextInt(); // this reads how many nodes will be given  
        Map<Integer, Node> nodes = new HashMap<>(); // stores all boxes  
        Set<Integer> children = new HashSet<>(); // to track child nodes  
  
        int firstNodeVal = -1; // will keep the very first node (main root)  
  
        // This will read all input lines and connect the boxes  
    }  
}
```

Output - binarytree_preorder (run) :

```
run:  
3  
1 2 -1  
2 -1 3  
3 -1 -1  
3 2 1  
BUILD SUCCESSFUL (total time: 1 second)
```

This screenshot is identical to the one above, showing the same Java code for binary tree post-order traversal. The code structure, variables, and output results are the same. The interface elements like menus, toolbars, and panes are also identical.

```
public class MyMain {  
    /*  
     * This function visits all boxes in postorder: left, right, root  
     */  
    public static void postOrder(Node root) {  
        // based on rule: visit left child, then right child, then current node  
        if (root != null) {  
            postOrder(root.left); // first visit left side  
            postOrder(root.right); // then visit right side  
            System.out.print(root.value + " "); // finally visit this box  
        }  
    }  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        int n = sc.nextInt(); // this reads how many nodes will be given  
        Map<Integer, Node> nodes = new HashMap<>(); // stores all boxes  
        Set<Integer> children = new HashSet<>(); // to track child nodes  
  
        int firstNodeVal = -1; // will keep the very first node (main root)  
  
        // This will read all input lines and connect the boxes  
    }  
}
```

Output - binarytree_preorder (run) :

```
run:  
4  
1 2 3  
2 4 -1  
3 -1 -1  
4 -1 -1  
4 2 3 1  
BUILD SUCCESSFUL (total time: 1 second)
```