

Write a Java Program to traverse a binary tree using PreOrder traversal without recursion

```
import java.util.Stack;

public class Main {

    public static void main(String[] args) throws Exception {

        BinaryTree bt = BinaryTree.create();

        System.out.println("printing nodes of a binary tree in preOrder using recursion");

        bt.preOrderWithoutRecursion();

    }

}

class BinaryTree {

    static class TreeNode {

        String data;

        TreeNode left, right;

        TreeNode(String value) {

            this.data = value; left = right = null;

        }

        boolean isLeaf() {

            return left == null ? right == null : false;

        }

    }

    TreeNode root;

    public void preOrderWithoutRecursion() {

        Stack<TreeNode> nodes = new Stack<>();

        nodes.push(root);

        while (!nodes.isEmpty()) {

            TreeNode current = nodes.pop();

            System.out.printf("%s ", current.data);

            if (current.right != null) {

                nodes.push(current.right);

            }

            if (current.left != null) {

                nodes.push(current.left);

            }

        }

    }

}
```

```
}}}
```

```
public static BinaryTree create() {
```

```
    BinaryTree tree = new BinaryTree();
```

```
    TreeNode root = new TreeNode("a");
```

```
    tree.root = root;
```

```
    tree.root.left = new TreeNode("b");
```

```
    tree.root.left.left = new TreeNode("c");
```

```
    tree.root.left.right = new TreeNode("d");
```

```
    tree.root.right = new TreeNode("e");
```

```
    tree.root.right.right = new TreeNode("f");
```

```
    return tree;
```

```
}}
```

The screenshot shows a web browser window with the URL `jdoodle.com/online-java-compiler/`. The page displays a Java program that creates a binary tree and prints its nodes in pre-order using recursion. The code is as follows:

```
1  System.out.println("printing nodes of a binary tree in preOrder using recursion");
2  bt.preOrderWithoutRecursion();
3  }
4  }
5
6  class BinaryTree {
7      static class TreeNode {
8          String data;
9          TreeNode left, right;
10         TreeNode(String value) {
11             this.data = value; left = right = null;
12         }
13     }
14     boolean isLeaf() {
15         return left == null & right == null : false;
16     }
17     TreeNode root;
18     public void preOrderWithoutRecursion() {
19         Stack<TreeNode> nodes = new Stack<>();
20         nodes.push(root);
21         while (!nodes.isEmpty()) {
22             TreeNode current = nodes.pop();
23             System.out.printf("%s ", current.data);
24             if (current.right != null) {
25                 nodes.push(current.right);
26             }
27             if (current.left != null) {
28                 nodes.push(current.left);
29             }
30         }
31     }
32     public static BinaryTree create() {
33         BinaryTree tree = new BinaryTree();
34         TreeNode root = new TreeNode("a");
35         tree.root = root;
36         tree.root.left = new TreeNode("b");
37         tree.root.left.left = new TreeNode("c");
38         tree.root.left.right = new TreeNode("d");
39         tree.root.right = new TreeNode("e");
40         tree.root.right.right = new TreeNode("f");
41         return tree;
42     }
43 }
44
45 }
```

The right sidebar shows the execution configuration: JDK 11.0.4, Stdin Inputs (empty), Interactive (unchecked), and Command Line Arguments (empty). The **Execute** button has been clicked. The **Result** section shows the output: `printing nodes of a binary tree in preOrder using recursion` followed by `a b c d e f`. The execution statistics are: CPU Time: 0.19 sec(s), Memory: 33336 kilobyte(s), compiled and executed in 0.862 sec(s).

Sponsored: [Try Azure free](#) Build AI apps with just a few lines of code. [Learn More](#)