**Solution@2-11-23**

## 258. Add Digits

```
import java.util.Scanner;

public class AddDigits {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter an integer: ");
        int num = scanner.nextInt();

        int result = addDigits(num);
        System.out.println("The result of adding digits is: " + result);

        scanner.close();
    }

    public static int addDigits(int num) {
        while (num >= 10) {
            int sum = 0;
            while (num > 0) {
                sum += num % 10;
                num /= 10;
            }
            num = sum;
        }
        return num;
    }
}
```

**Output:-**

```
java -cp /tmp/kbMnriXAVq AddDigits
Enter an integer: 1235
The result of adding digits is: 2
```

**Without Loop**

```
import java.util.Scanner;

public class AddDigits {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter an integer: ");
        int num = scanner.nextInt();

        int result = addDigits(num);
        System.out.println("The result of adding digits is: " + result);

        scanner.close();
    }

    public static int addDigits(int num) {
        if (num == 0) {
            return 0;
        }
        return 1 + (num - 1) % 9;
    }
}
```

**Output:**

```
java -cp /tmp/kbMnriXAVq AddDigits
Enter an integer: 1235
```

The result of adding digits is: 2

## 257. Binary Tree Paths

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int x) {
        val = x;
    }
}

public class BinaryTreePaths {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the elements of the binary tree separated by spaces (e.g., 1 2 3 4 5 6 7): ");
        String input = scanner.nextLine();

        String[] elements = input.split(" ");
        TreeNode root = buildTree(elements, 0);

        List<String> paths = binaryTreePaths(root);
```

```
    System.out.println("Root-to-Leaf Paths:");
    for (String path : paths) {
        System.out.println(path);
    }

    scanner.close();
}

public static List<String> binaryTreePaths(TreeNode root) {
    List<String> paths = new ArrayList<>();
    if (root != null) {
        findPaths(root, "", paths);
    }
    return paths;
}

private static void findPaths(TreeNode node, String path, List<String> paths) {
    if (node.left == null && node.right == null) {
        paths.add(path + node.val);
    }

    if (node.left != null) {
        findPaths(node.left, path + node.val + "->", paths);
    }

    if (node.right != null) {
        findPaths(node.right, path + node.val + "->", paths);
    }
}
```

```
    private static TreeNode buildTree(String[] elements, int index) {
        if (index >= elements.length) {
            return null;
        }

        if (elements[index].equals("null")) {
            return null;
        }

        TreeNode node = new TreeNode(Integer.parseInt(elements[index]));
        node.left = buildTree(elements, 2 * index + 1);
        node.right = buildTree(elements, 2 * index + 2);

        return node;
    }
}
```

**Output:-**

**PS C:\Users\Ajeet\Desktop\java1> javac BinaryTreePaths.java**

**PS C:\Users\Ajeet\Desktop\java1> java BinaryTreePaths**

**Enter the elements of the binary tree separated by spaces (e.g., 1 2 3 4 5 6 7): 1 4 5 8 7**

**Root-to-Leaf Paths:**

**1->4->8**

**1->4->7**

**1->5**