RECURSION USING STACK

```
package recursion;
import java.util.Stack;
public class Factorial {

        // Recursive factorial function
    public static int recursiveFactorial(int n) {
        if (n == 0 || n == 1) {
            return 1;
        } else {
            return n * recursiveFactorial(n - 1);
        }
    }

    // Factorial with stack function
    public static int factorialWithStack(int n) {
        // Create a stack to simulate recursion
        Stack<Integer> stack = new Stack<>();
        stack.push(n);  // Push the initial value onto the stack
        int result = 1; // Initialize the result variable to 1

        // Iterate until the stack is empty
        while (!stack.isEmpty()) {
            int num = stack.pop(); // Pop a number from the stack

            // Multiply the result by the popped number
            result *= num;

            // If the popped number is greater than 1, push (num - 1) onto the
stack
            if (num > 1) {
                stack.push(num - 1);
            }
        }
        return result; // Return the factorial result
    }

    // Main method to test the recursive and stack-based factorial functions
    public static void main(String[] args) {
        int num = 5; // Number for which factorial is calculated
```

```
            // Calculate and print factorial using recursion
            System.out.println("Factorial of " + num + " using recursion: " +
recursiveFactorial(num));

            // Calculate and print factorial using stack
            System.out.println("Factorial of " + num + " using stack: " +
factorialWithStack(num));
        }
    }
```