**public** **class** SwapNos {

**public** **static** **void** main(String[] args) {

    System.out.println("Calling swap function with inputs 2 & 3");

    swap(2,3);

    System.out.println("Calling swap function with inputs -3 & 5");

    swap(-3,5);

  }

**private** **static** **void** swap(**int** x, **int** y) {

    System.out.println("values before swap:" + x + " and " + y);

    // swap logic

    x = x + y;

    y = x - y;

    x = x - y;

    System.out.println("values after swap:" + x + " and " + y);

  }

}

**public** **class** ReverseNumber {

**public** **static** **void** main(String[] args) {

**int** num = 10025;

    System.out.println("Input - " + num + " Output:" +

reverseNo(num));

  }

**public** **static** **int** reverseNo(**int** number) {

**int** reversed = 0;

**while**(number != 0) {

**int** digit = number % 10;

          reversed = reversed \* 10 + digit;

          number /= 10;

    }

**return**  reversed;

  }

}

**public** **class** Factorial {

**public** **static** **void** main(String[] args) {

    System.out.println("Factorial of 5 using loop is:" + factorialWithLoop(5));

    System.out.println("Factorial of 10 using recursion is:" + factorialWithRecursion(10));

    System.out.println("Factorial of negative number -100 is:" + factorialWithLoop(-100));

  }

**public** **static** **long** factorialWithLoop(**int** n) {

**if**(n < 0) {

      System.out.println("Negative nos can't have factorial");

**return** -9999;

    }

**long** fact = 1;

**for** (**int** i = 2; i <= n; i++) {

      fact = fact \* i;

    }

**return** fact;

  }

**public** **static** **long** factorialWithRecursion(**int** n) {

**if**(n < 0) {

      System.out.println("Negative nos can't have factorial");

**return** -9999;

    }

**if** (n <= 2) {

**return** n;

    }

**return** n \* factorialWithRecursion(n - 1);

  }

}

**import** java.util.Stack;

**public** **class** BalancedParanthesis {

**public** **static** **void** main(String[] args) {

**final** String input1 = "{()}";

    System.out.println("Checking balanced paranthesis for input:" + input1);

**if** (isBalanced(input1)) {

      System.out.println("Given String is balanced");

    } **else** {

      System.out.println("Given String is not balanced");

    }

  }

  /\*\*

   \* function to check if a string has balanced parentheses or not

   \* @param input\_string the input string

   \* @return if the string has balanced parentheses or not

   \*/

**private** **static** **boolean** isBalanced(String input\_string) {

    Stack<Character> stack = **new** Stack<>();

**for** (**int** i = 0; i < input\_string.length(); i++) {

**switch** (input\_string.charAt(i)) {

**case** '[':

**case** '(':

**case** '{':

          stack.push(input\_string.charAt(i));

**break**;

**case** ']':

**if** (stack.empty() || !stack.pop().equals('[')) {

**return** **false**;

          }

**break**;

**case** '}':

**if** (stack.empty() || !stack.pop().equals('{')) {

**return** **false**;

          }

**break**;

**case** ')':

**if** (stack.empty() || !stack.pop().equals('(')) {

**return** **false**;

          }

**break**;

      }

    }

**return** stack.empty();

  }

}

* [There are five popular frameworks in test automation:](https://www.softwaretestinghelp.com/automation-testing-tutorial-5/#There_are_five_popular_frameworks_in_test_automation)
* [#1. Linear Framework:](https://www.softwaretestinghelp.com/automation-testing-tutorial-5/#1_Linear_Framework)
* [#2. Modularity Framework:](https://www.softwaretestinghelp.com/automation-testing-tutorial-5/#2_Modularity_Framework)
* [#3. Data Driven Framework:](https://www.softwaretestinghelp.com/automation-testing-tutorial-5/#3_Data_Driven_Framework)
* [#4. Keyword-Driven Framework:](https://www.softwaretestinghelp.com/automation-testing-tutorial-5/#4_Keyword-Driven_Framework)
* [#5. Hybrid Test Automation Framework:](https://www.softwaretestinghelp.com/automation-testing-tutorial-5/#5_Hybrid_Test_Automation_Framework)