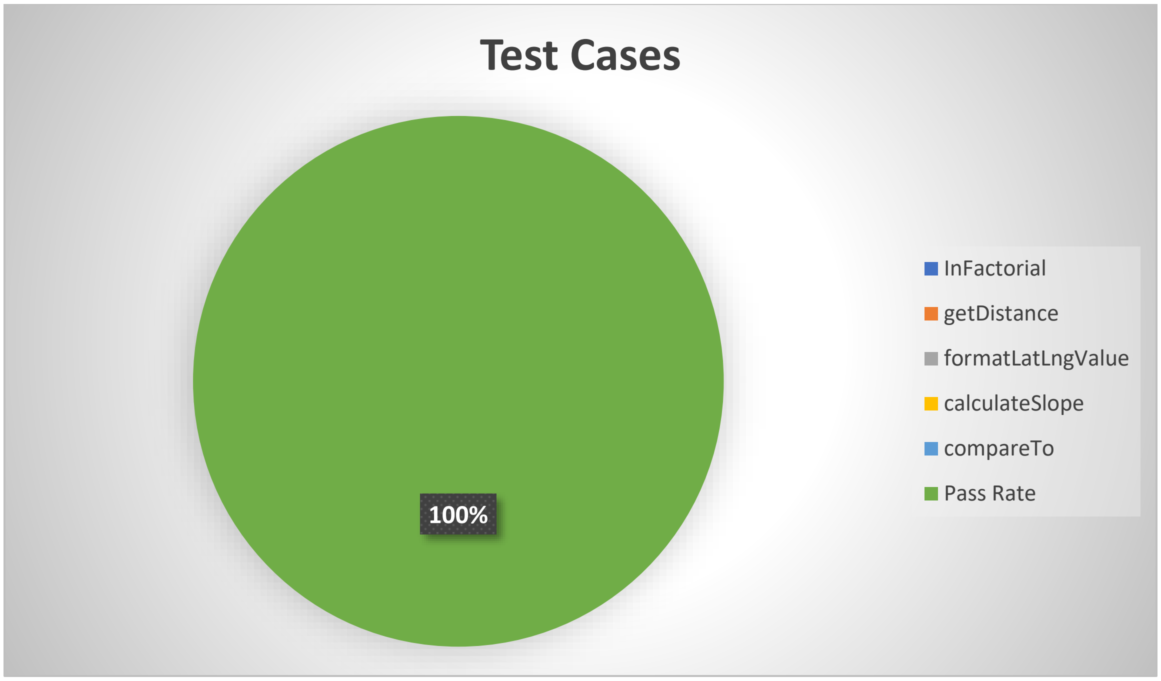
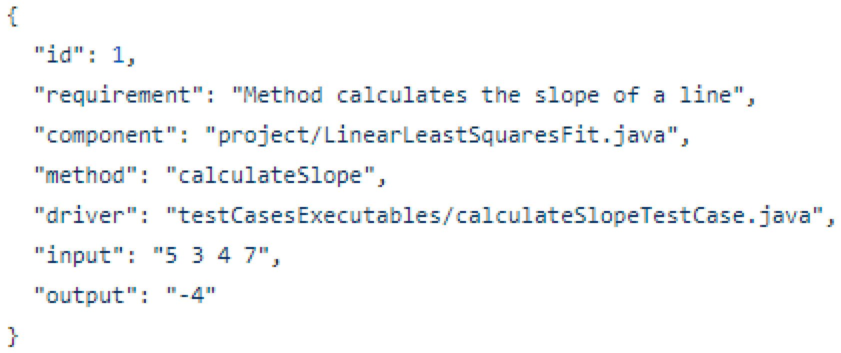


READ JSON FILE

```
# READ JSON FILE

# This method will read a JSON file and return a JSON object

# Input: file path to JSON file
# Output: JSON object

def readJsonAtLocation(filePath):

    # Change the directory to the given path
    os.chdir(filePath[0:filePath.rindex("/")])

    # Split file path
    splitFilePath = filePath.split("/")

    # Parse out the name of the file
    fileName = splitFilePath[len(splitFilePath)-1]

    jsonFile = open(fileName)

    jsonData = json.load(jsonFile)

    # Change the directory back to the way it was...
    os.chdir("../")

    return jsonData
```

The code provided above reads a JSON file which uses human-readable text to store and transmit data objects consisting of attribute value pairs and array data types.

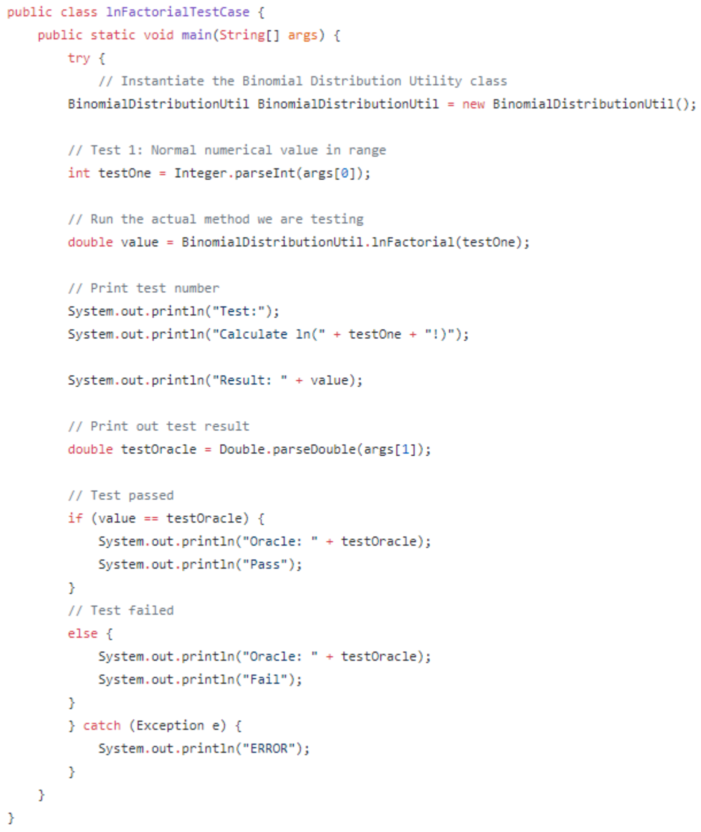
LNFACTORIAL JSON FILE



```
{
  "id": 1,
  "requirement": "Method calculates the slope of a line",
  "component": "project/LinearLeastSquaresFit.java",
  "method": "calculateSlope",
  "driver": "testCasesExecutables/calculateSlopeTestCase.java",
  "input": "5 3 4 7",
  "output": "-4"
}
```

It uses the JSON file to run and execute each test case.

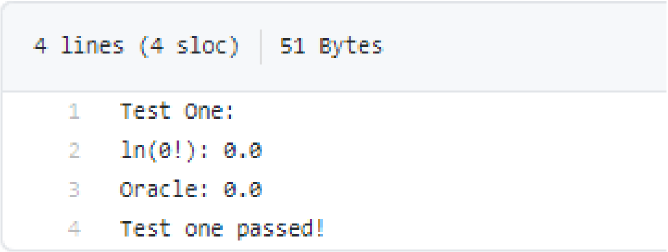
TESTCASE EXECUTABLE



```
public class lnFactorialTestCase {
    public static void main(String[] args) {
        try {
            // Instantiate the Binomial Distribution Utility class
        BinomialDistributionUtil BinomialDistributionUtil = new BinomialDistributionUtil();

        // Test 1: Normal numerical value in range
        int testOne = Integer.parseInt(args[0]);

        // Run the actual method we are testing
        double value = BinomialDistributionUtil.lnFactorial(testOne);

        // Print test number
        System.out.println("Test:");
        System.out.println("Calculate ln(" + testOne + "!)");

        System.out.println("Result: " + value);

        // Print out test result
        double testOracle = Double.parseDouble(args[1]);

        // Test passed
        if (value == testOracle) {
            System.out.println("Oracle: " + testOracle);
            System.out.println("Pass");
        }
        // Test failed
        else {
            System.out.println("Oracle: " + testOracle);
            System.out.println("Fail");
        }
        } catch (Exception e) {
            System.out.println("ERROR");
        }
    }
}
```

The results will be collected and compared with the expected results.

TESTCASE RSULTS

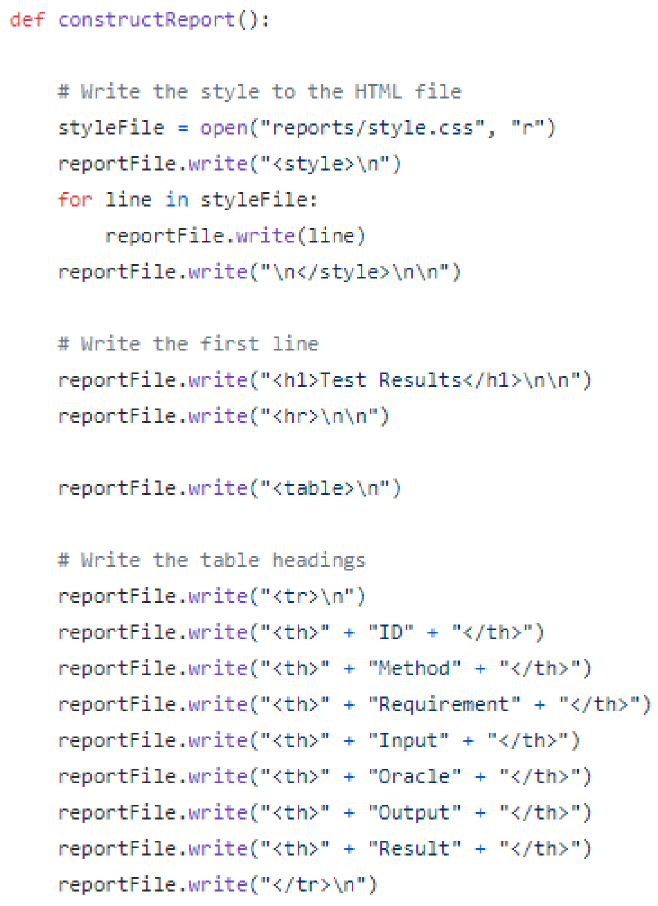


4 lines (4 sloc) | 51 Bytes

```
Test One:
ln(0!): 0.0
Oracle: 0.0
Test one passed!
```

After the test cases are ran the constructReport() method is called which combs though the temporary results files and constructs a final report as a HTML document.

constructReport() method

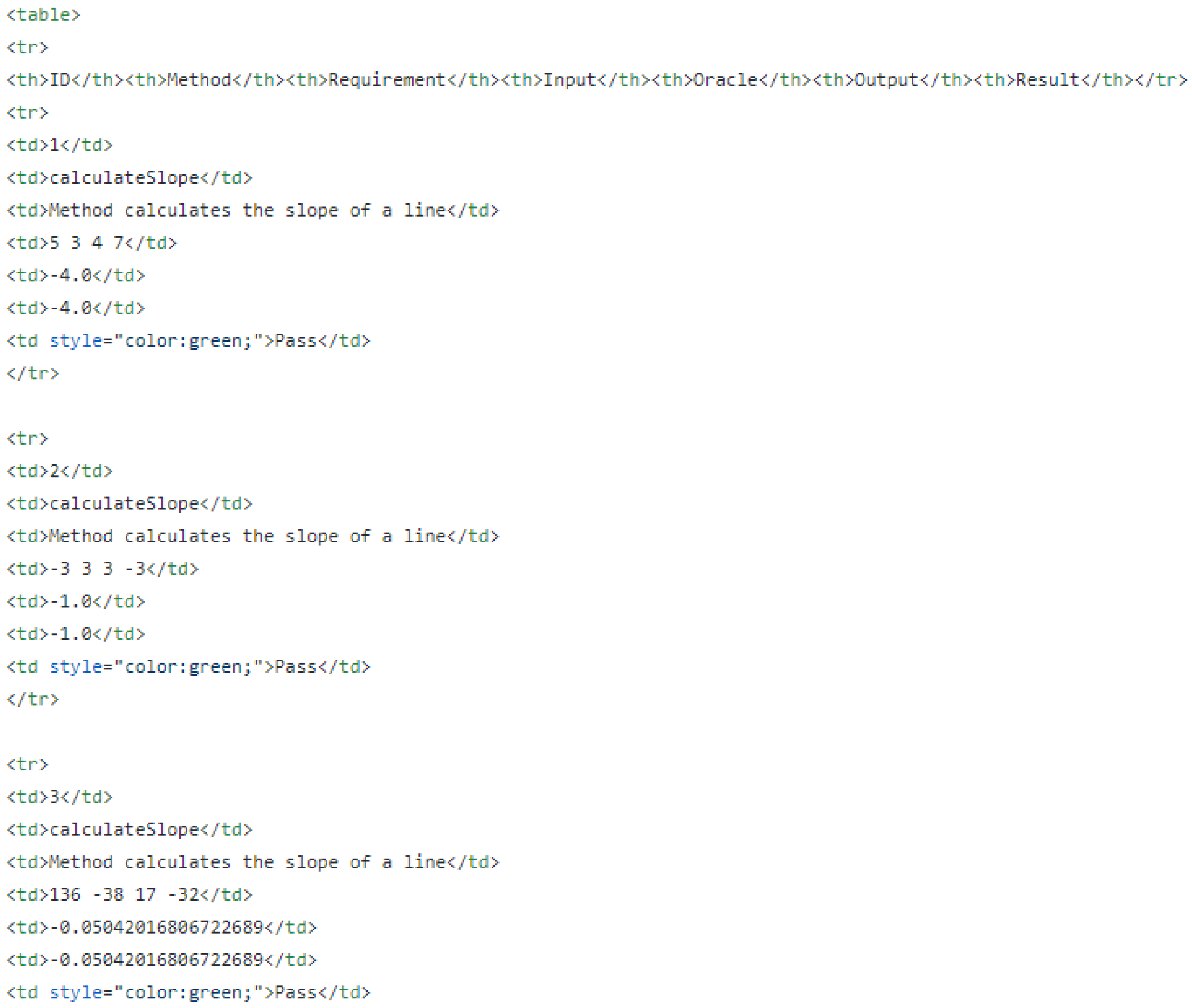


```
def constructReport():

    # Write the style to the HTML file
    styleFile = open("reports/style.css", "r")
    reportFile.write("<style>\n")
    for line in styleFile:
        reportFile.write(line)
    reportFile.write("\n</style>\n\n")

    # Write the first line
    reportFile.write("<h1>Test Results</h1>\n\n")
    reportFile.write("<hr>\n\n")

    reportFile.write("<table>\n")

    # Write the table headings
    reportFile.write("<tr>\n")
    reportFile.write("<th>" + "ID" + "</th>")
    reportFile.write("<th>" + "Method" + "</th>")
    reportFile.write("<th>" + "Requirement" + "</th>")
    reportFile.write("<th>" + "Input" + "</th>")
    reportFile.write("<th>" + "Oracle" + "</th>")
    reportFile.write("<th>" + "Output" + "</th>")
    reportFile.write("<th>" + "Result" + "</th>")
    reportFile.write("</tr>\n")
```

Team Term Project

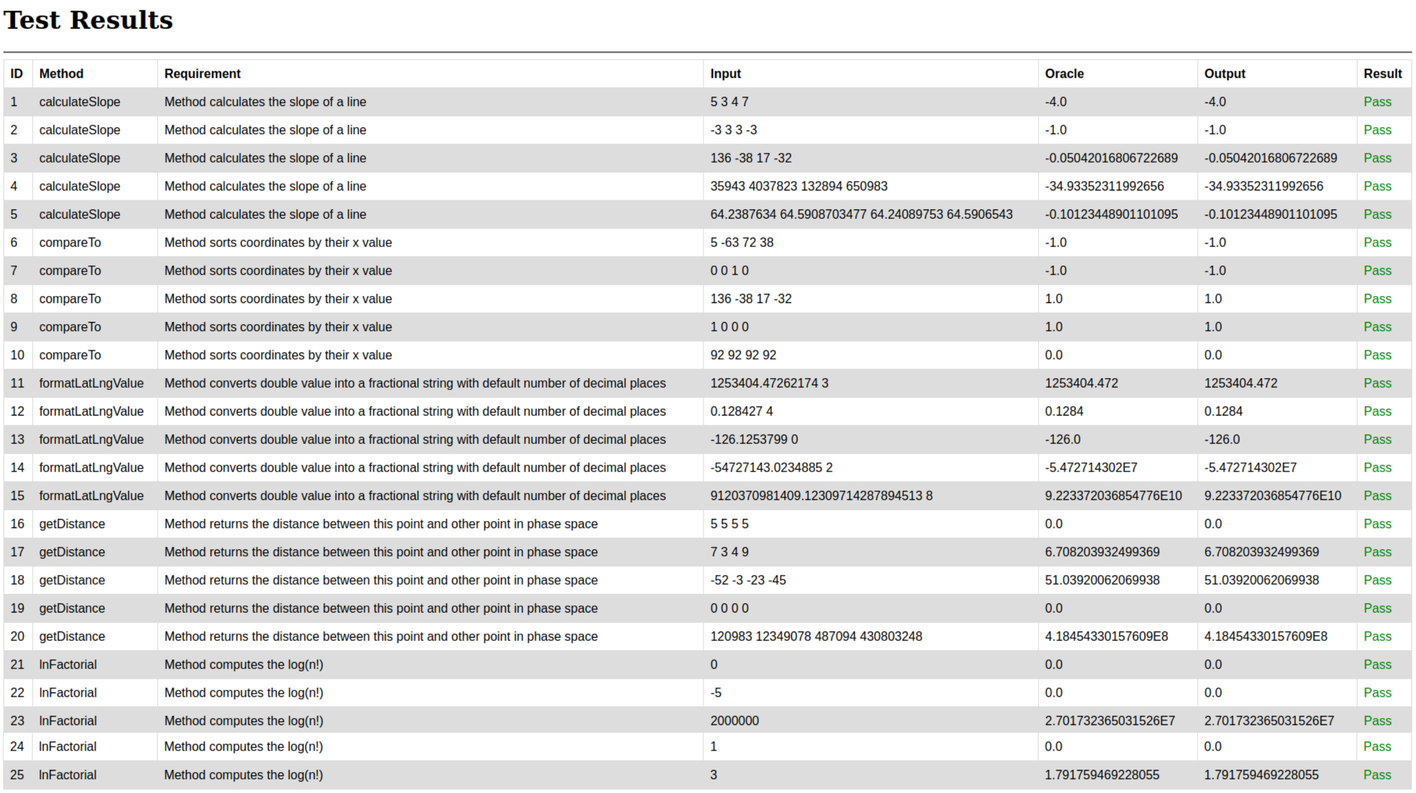
The following final report is constructed after all the test cases are ran

FINAL TEST REPORT



```
<table>
<tr>
<th>ID</th><th>Method</th><th>Requirement</th><th>Input</th><th>Oracle</th><th>Output</th><th>Result</th></tr>
<tr>
<td>1</td>
<td>calculateSlope</td>
<td>Method calculates the slope of a line</td>
<td>5 3 4 7</td>
<td>-4.0</td>
<td>-4.0</td>
<td style="color:green;">Pass</td>
</tr>

<tr>
<td>2</td>
<td>calculateSlope</td>
<td>Method calculates the slope of a line</td>
<td>-3 3 3 -3</td>
<td>-1.0</td>
<td>-1.0</td>
<td style="color:green;">Pass</td>
</tr>

<tr>
<td>3</td>
<td>calculateSlope</td>
<td>Method calculates the slope of a line</td>
<td>136 -38 17 -32</td>
<td>-0.05042016806722689</td>
<td>-0.05042016806722689</td>
<td style="color:green;">Pass</td>
```

Final Output



Test Results

ID	Method	Requirement	Input	Oracle	Output	Result
1	calculateSlope	Method calculates the slope of a line	5 3 4 7	-4.0	-4.0	Pass
2	calculateSlope	Method calculates the slope of a line	-3 3 3 -3	-1.0	-1.0	Pass
3	calculateSlope	Method calculates the slope of a line	136 -38 17 -32	-0.05042016806722689	-0.05042016806722689	Pass
4	calculateSlope	Method calculates the slope of a line	35943 4037823 132894 650983	-34.93352311992656	-34.93352311992656	Pass
5	calculateSlope	Method calculates the slope of a line	64.2387634 64.5908703477 64.24089753 64.5906543	-0.10123448901101095	-0.10123448901101095	Pass
6	compareTo	Method sorts coordinates by their x value	5 -63 72 38	-1.0	-1.0	Pass
7	compareTo	Method sorts coordinates by their x value	0 0 1 0	-1.0	-1.0	Pass
8	compareTo	Method sorts coordinates by their x value	136 -38 17 -32	1.0	1.0	Pass
9	compareTo	Method sorts coordinates by their x value	1 0 0 0	1.0	1.0	Pass
10	compareTo	Method sorts coordinates by their x value	92 92 92 92	0.0	0.0	Pass
11	formatLatLngValue	Method converts double value into a fractional string with default number of decimal places	1253404.47262174 3	1253404.472	1253404.472	Pass
12	formatLatLngValue	Method converts double value into a fractional string with default number of decimal places	0.128427 4	0.1284	0.1284	Pass
13	formatLatLngValue	Method converts double value into a fractional string with default number of decimal places	-126.1253799 0	-126.0	-126.0	Pass
14	formatLatLngValue	Method converts double value into a fractional string with default number of decimal places	-54727143.0234885 2	-5.472714302E7	-5.472714302E7	Pass
15	formatLatLngValue	Method converts double value into a fractional string with default number of decimal places	9120370981409.12309714287894513 8	9.223372036854776E10	9.223372036854776E10	Pass
16	getDistance	Method returns the distance between this point and other point in phase space	5 5 5 5	0.0	0.0	Pass
17	getDistance	Method returns the distance between this point and other point in phase space	7 3 4 9	6.708203932499369	6.708203932499369	Pass
18	getDistance	Method returns the distance between this point and other point in phase space	-52 -3 -23 -45	51.03920062069938	51.03920062069938	Pass
19	getDistance	Method returns the distance between this point and other point in phase space	0 0 0 0	0.0	0.0	Pass
20	getDistance	Method returns the distance between this point and other point in phase space	120983 12349078 487094 430803248	4.18454330157609E8	4.18454330157609E8	Pass
21	lnFactorial	Method computes the log(n!)	0	0.0	0.0	Pass
22	lnFactorial	Method computes the log(n!)	-5	0.0	0.0	Pass
23	lnFactorial	Method computes the log(n!)	2000000	2.701732365031526E7	2.701732365031526E7	Pass
24	lnFactorial	Method computes the log(n!)	1	0.0	0.0	Pass
25	lnFactorial	Method computes the log(n!)	3	1.791759469228055	1.791759469228055	Pass