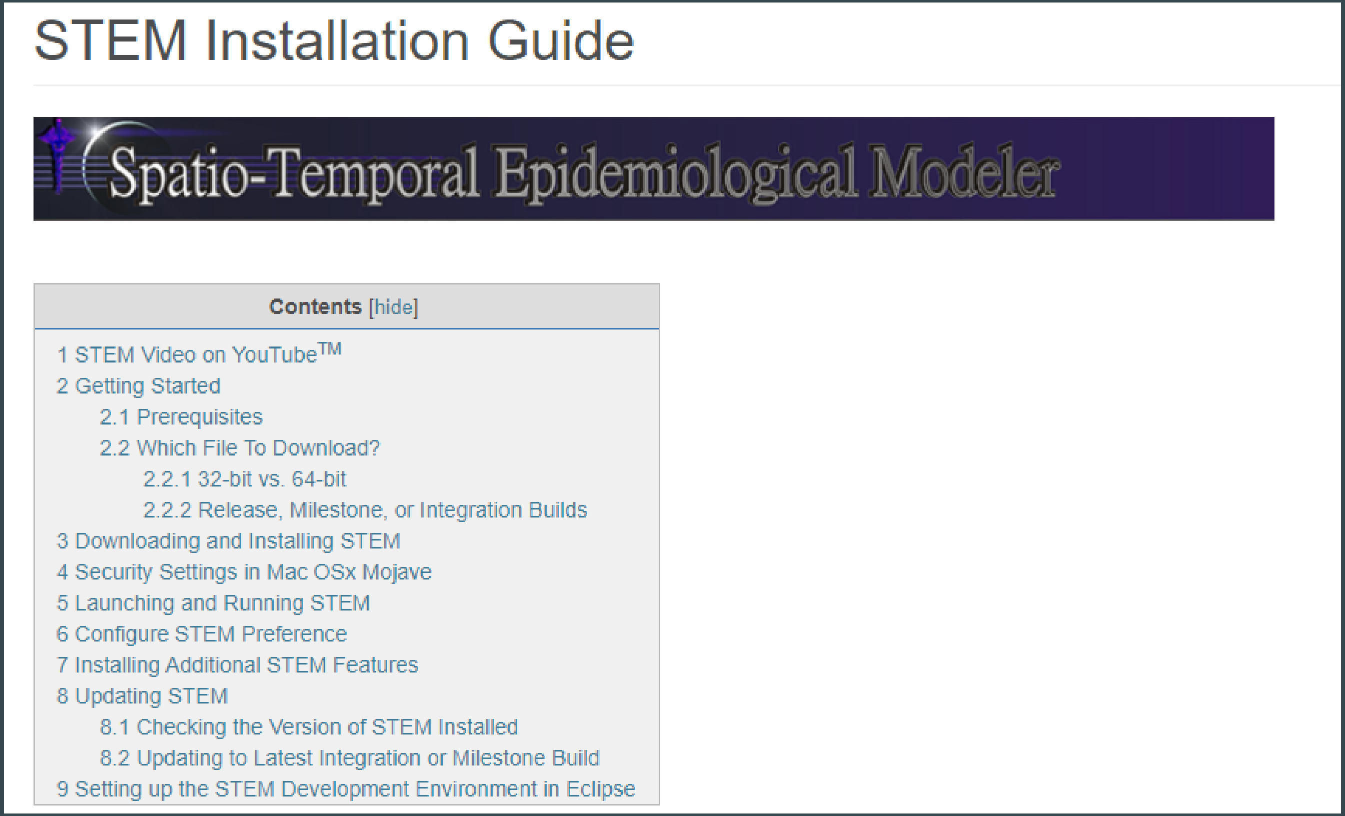
STEM (Spatio-temporal Epidemiological Modeler)

•••

Katherine Sweeney, Ashanti Long, Clare Clever

What is STEM?

Why We Chose STEM



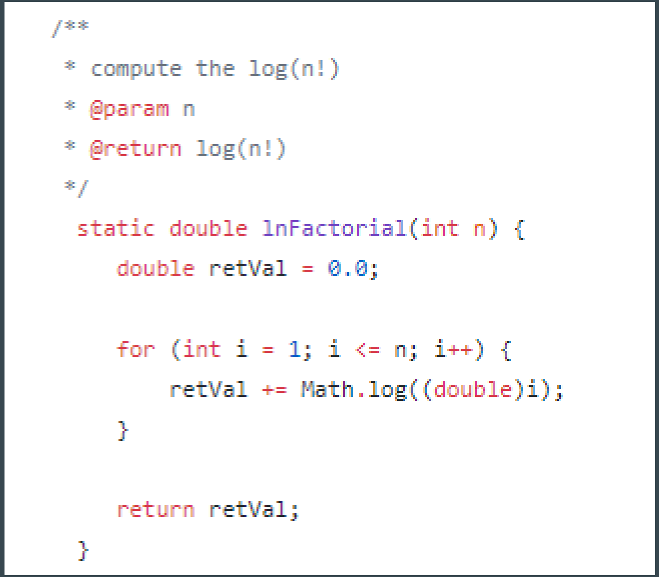
STEM Installation Guide

Spatio-Temporal Epidemiological Modeler

Contents [hide]

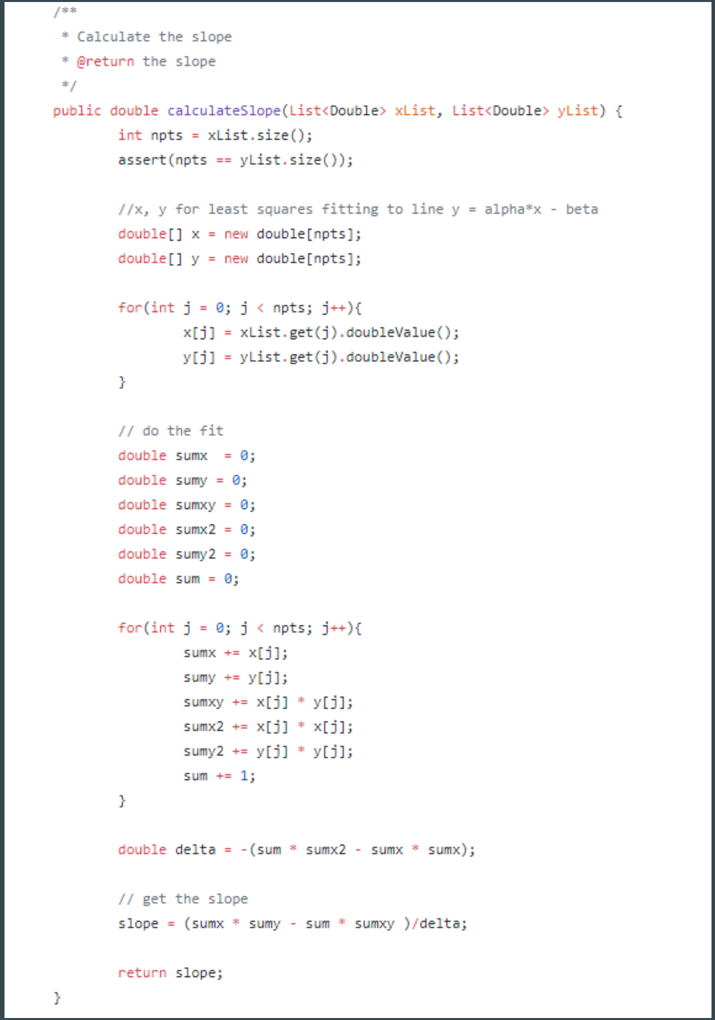
Method Selection

lnFactorial()



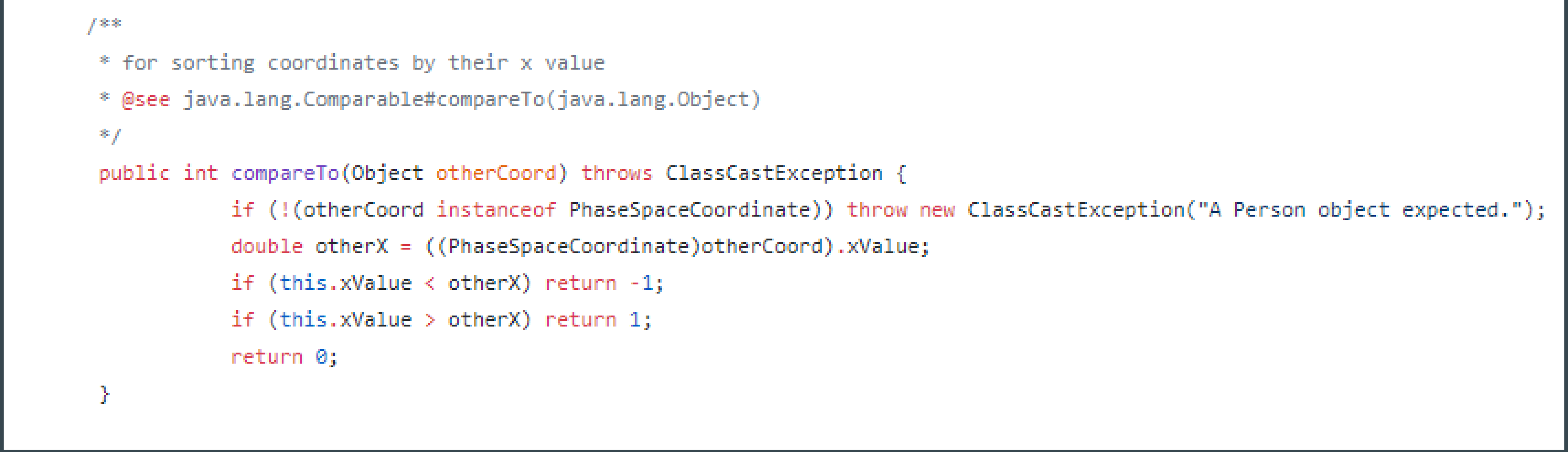
```
/**
 * compute the log(n!)
 * @param n
 * @return log(n!)
 */
  static double lnFactorial(int n) {
     double retVal = 0.0;

     for (int i = 1; i <= n; i++) {
         retVal += Math.log((double)i);
     }

     return retVal;
  }
```

calculateSlope()



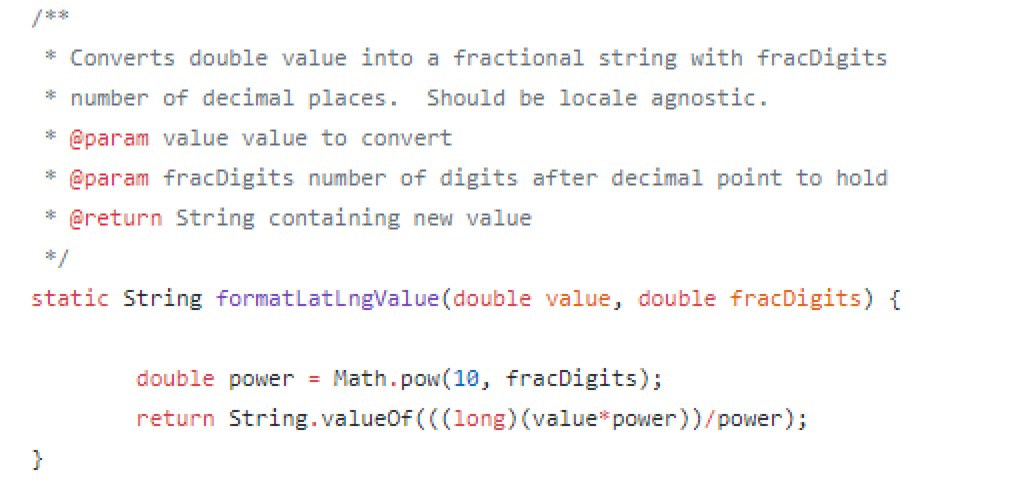
```java
/**
 * Calculate the slope
 * @return the slope
 */
public double calculateSlope(List<Double> xList, List<Double> yList) {
        int npts = xList.size();
        assert(npts == yList.size());

        //x, y for least squares fitting to line y = alpha*x - beta
        double[] x = new double[npts];
        double[] y = new double[npts];

        for(int j = 0; j < npts; j++){
                x[j] = xList.get(j).doubleValue();
                y[j] = yList.get(j).doubleValue();
        }

        // do the fit
        double sumx  = 0;
        double sumy = 0;
        double sumxy = 0;
        double sumx2 = 0;
        double sumy2 = 0;
        double sum = 0;

        for(int j = 0; j < npts; j++){
                sumx += x[j];
                sumy += y[j];
                sumxy += x[j] * y[j];
                sumx2 += x[j] * x[j];
                sumy2 += y[j] * y[j];
                sum += 1;
        }

        double delta = -(sum * sumx2 - sumx * sumx);

        // get the slope
        slope = (sumx * sumy - sum * sumxy )/delta;

        return slope;
}
```

compareTo()



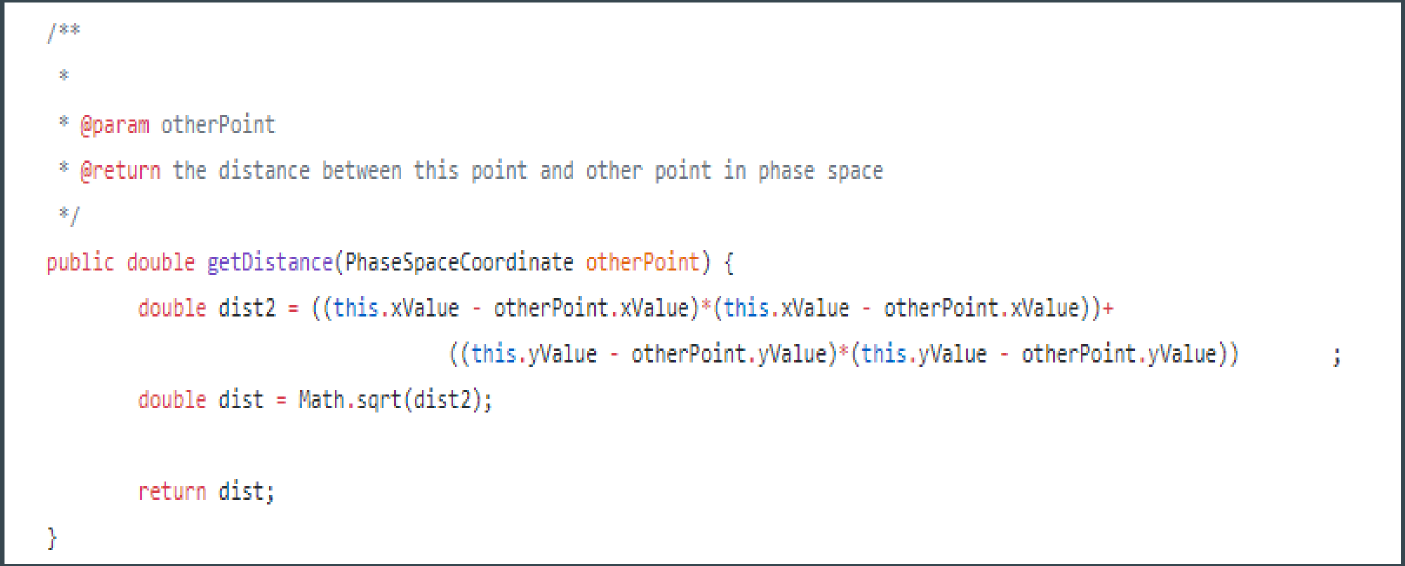
```
/**
 * for sorting coordinates by their x value
 * @see java.lang.Comparable#compareTo(java.lang.Object)
 */
 public int compareTo(Object otherCoord) throws ClassCastException {
            if (!(otherCoord instanceof PhaseSpaceCoordinate)) throw new ClassCastException("A Person object expected.");
            double otherX = ((PhaseSpaceCoordinate)otherCoord).xValue;
            if (this.xValue < otherX) return -1;
            if (this.xValue > otherX) return 1;
            return 0;
 }
```

formatLatLngValue()



```
/**
 * Converts double value into a fractional string with fracDigits
 * number of decimal places.  Should be locale agnostic.
 * @param value value to convert
 * @param fracDigits number of digits after decimal point to hold
 * @return String containing new value
 */
static String formatLatLngValue(double value, double fracDigits) {

        double power = Math.pow(10, fracDigits);
        return String.valueOf(((long)(value*power))/power);
}
```

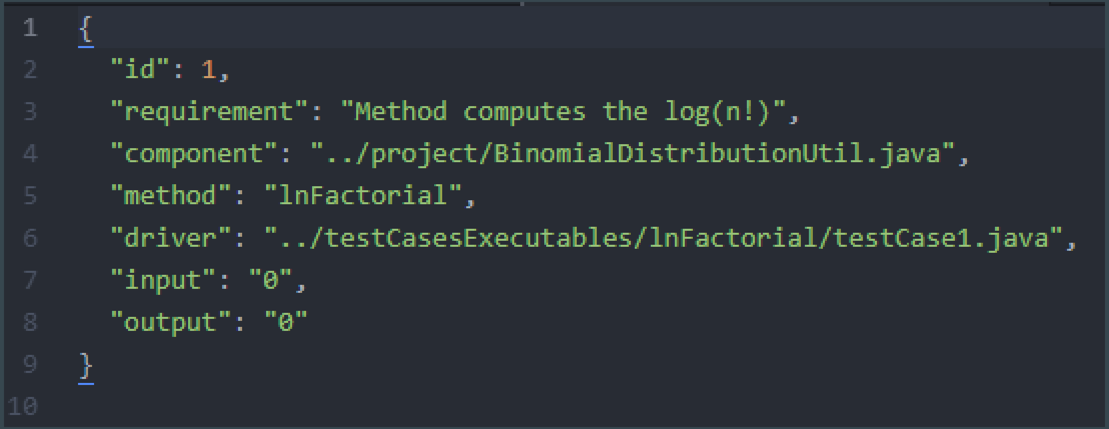
getDistance()



```java
/**
 *
 * @param otherPoint
 * @return the distance between this point and other point in phase space
 */
public double getDistance(PhaseSpaceCoordinate otherPoint) {
        double dist2 = ((this.xValue - otherPoint.xValue)*(this.xValue - otherPoint.xValue))+
                                   ((this.yValue - otherPoint.yValue)*(this.yValue - otherPoint.yValue))       ;
        double dist = Math.sqrt(dist2);

        return dist;
}
```

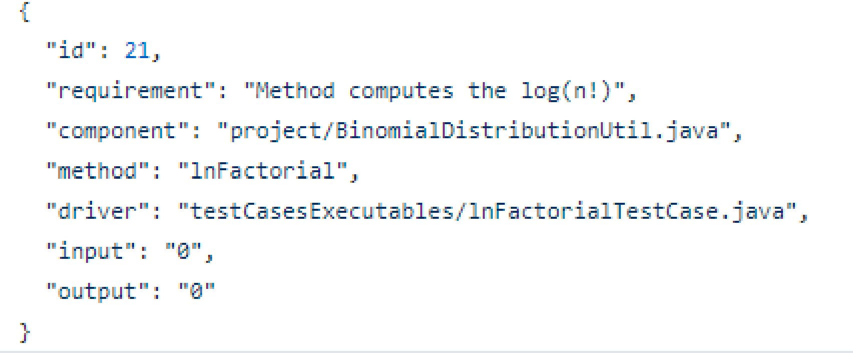
Test Case Progression

Initial



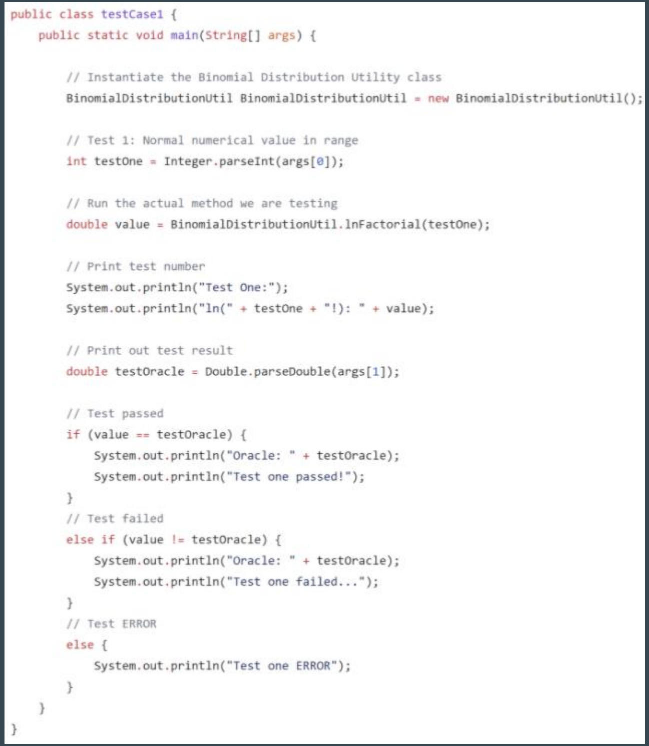
```
{
  "id": 1,
  "requirement": "Method computes the log(n!)",
  "component": "../project/BinomialDistributionUtil.java",
  "method": "lnFactorial",
  "driver": "../testCasesExecutables/lnFactorial/testCase1.java",
  "input": "0",
  "output": "0"
}
```

Final



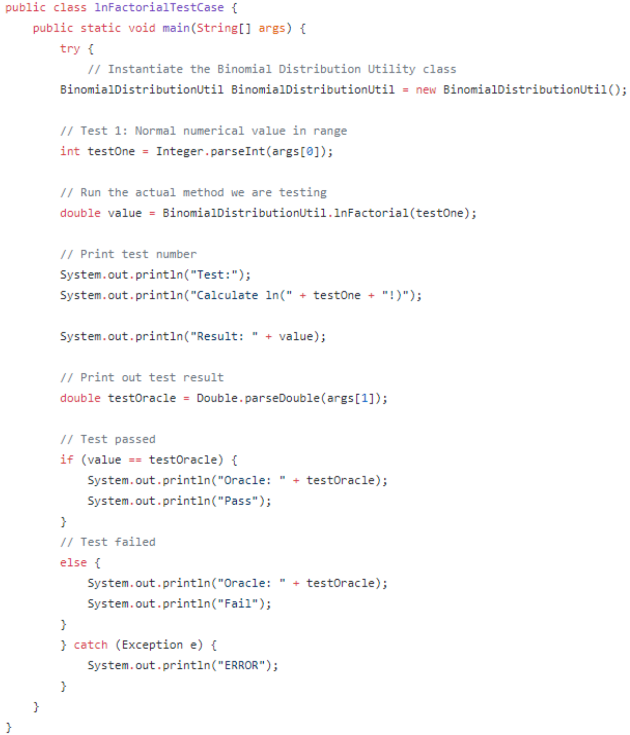
```
{
  "id": 21,
  "requirement": "Method computes the log(n!)",
  "component": "project/BinomialDistributionUtil.java",
  "method": "lnFactorial",
  "driver": "testCasesExecutables/lnFactorialTestCase.java",
  "input": "0",
  "output": "0"
}
```

Bad Driver



```
public class testCase1 {
    public static void main(String[] args) {

        // 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 One:");
        System.out.println("ln(" + testOne + "!): " + value);

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

        // Test passed
        if (value == testOracle) {
            System.out.println("Oracle: " + testOracle);
            System.out.println("Test one passed!");
        }
        // Test failed
        else if (value != testOracle) {
            System.out.println("Oracle: " + testOracle);
            System.out.println("Test one failed...");
        }
        // Test ERROR
        else {
            System.out.println("Test one ERROR");
        }
    }
}
```

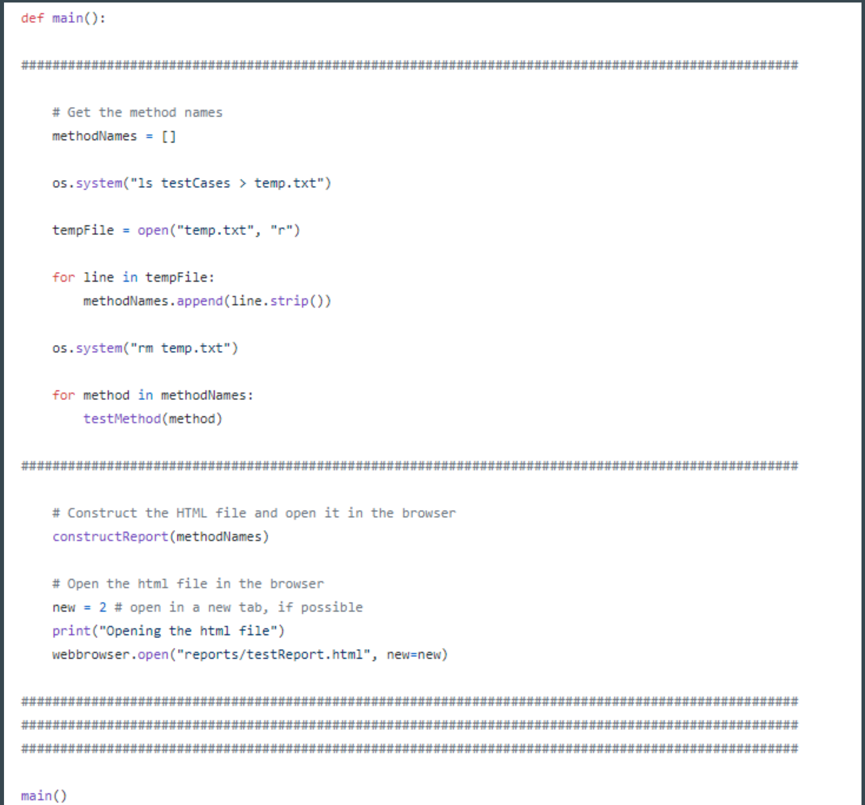
Improved Driver



```
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");
        }
    }
}
```

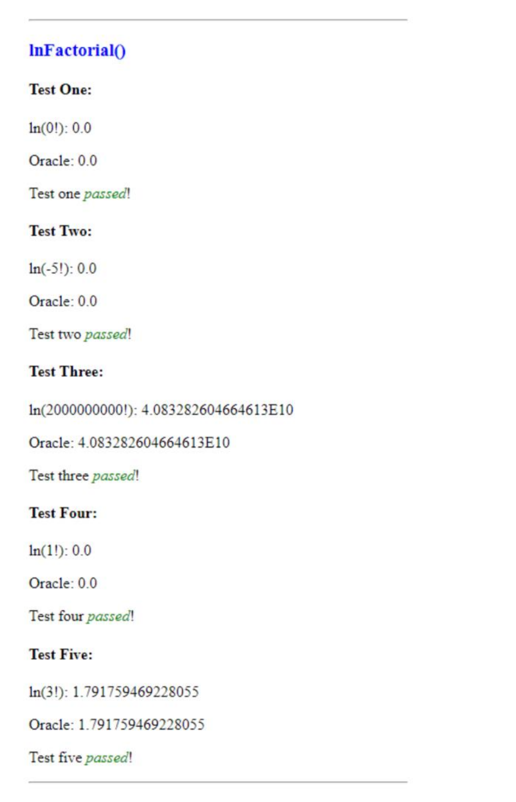
Script Mistakes

Incorrect Script Attempt



```
def main():

####################################################################################################

    # Get the method names
    methodNames = []

    os.system("ls testCases > temp.txt")

    tempFile = open("temp.txt", "r")

    for line in tempFile:
        methodNames.append(line.strip())

    os.system("rm temp.txt")

    for method in methodNames:
        testMethod(method)

####################################################################################################

    # Construct the HTML file and open it in the browser
    constructReport(methodNames)

    # Open the html file in the browser
    new = 2 # open in a new tab, if possible
    print("Opening the html file")
    webbrowser.open("reports/testReport.html", new=new)

####################################################################################################
####################################################################################################
####################################################################################################

main()
```

Test Result Report Progression



lnFactorial()

Test One:

ln(0!): 0.0

Oracle: 0.0

Test one *passed*!

Test Two:

ln(-5!): 0.0

Oracle: 0.0

Test two *passed*!

Test Three:

ln(2000000000!): 4.083282604664613E10

Oracle: 4.083282604664613E10

Test three *passed*!

Test Four:

ln(1!): 0.0

Oracle: 0.0

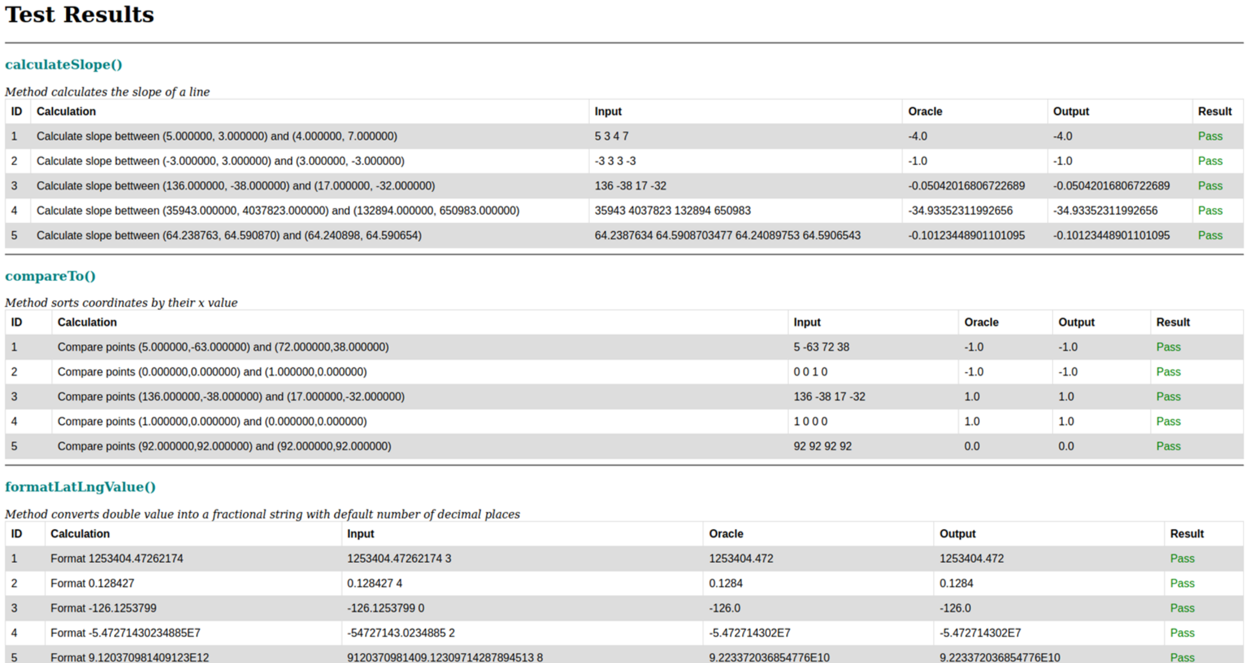
Test four *passed*!

Test Five:

ln(3!): 1.791759469228055

Oracle: 1.791759469228055

Test five *passed*!



Test Results

calculateSlope()

Method calculates the slope of a line

ID	Calculation	Input	Oracle	Output	Result
1	Calculate slope bettween (5.000000, 3.000000) and (4.000000, 7.000000)	5 3 4 7	-4.0	-4.0	Pass
2	Calculate slope bettween (-3.000000, 3.000000) and (3.000000, -3.000000)	-3 3 3 -3	-1.0	-1.0	Pass
3	Calculate slope bettween (136.000000, -38.000000) and (17.000000, -32.000000)	136 -38 17 -32	-0.05042016806722689	-0.05042016806722689	Pass
4	Calculate slope bettween (35943.000000, 4037823.000000) and (132894.000000, 650983.000000)	35943 4037823 132894 650983	-34.93352311992656	-34.93352311992656	Pass
5	Calculate slope bettween (64.238763, 64.590870) and (64.240898, 64.590654)	64.2387634 64.5908703477 64.24089753 64.5906543	-0.10123448901101095	-0.10123448901101095	Pass

compareTo()

Method sorts coordinates by their x value

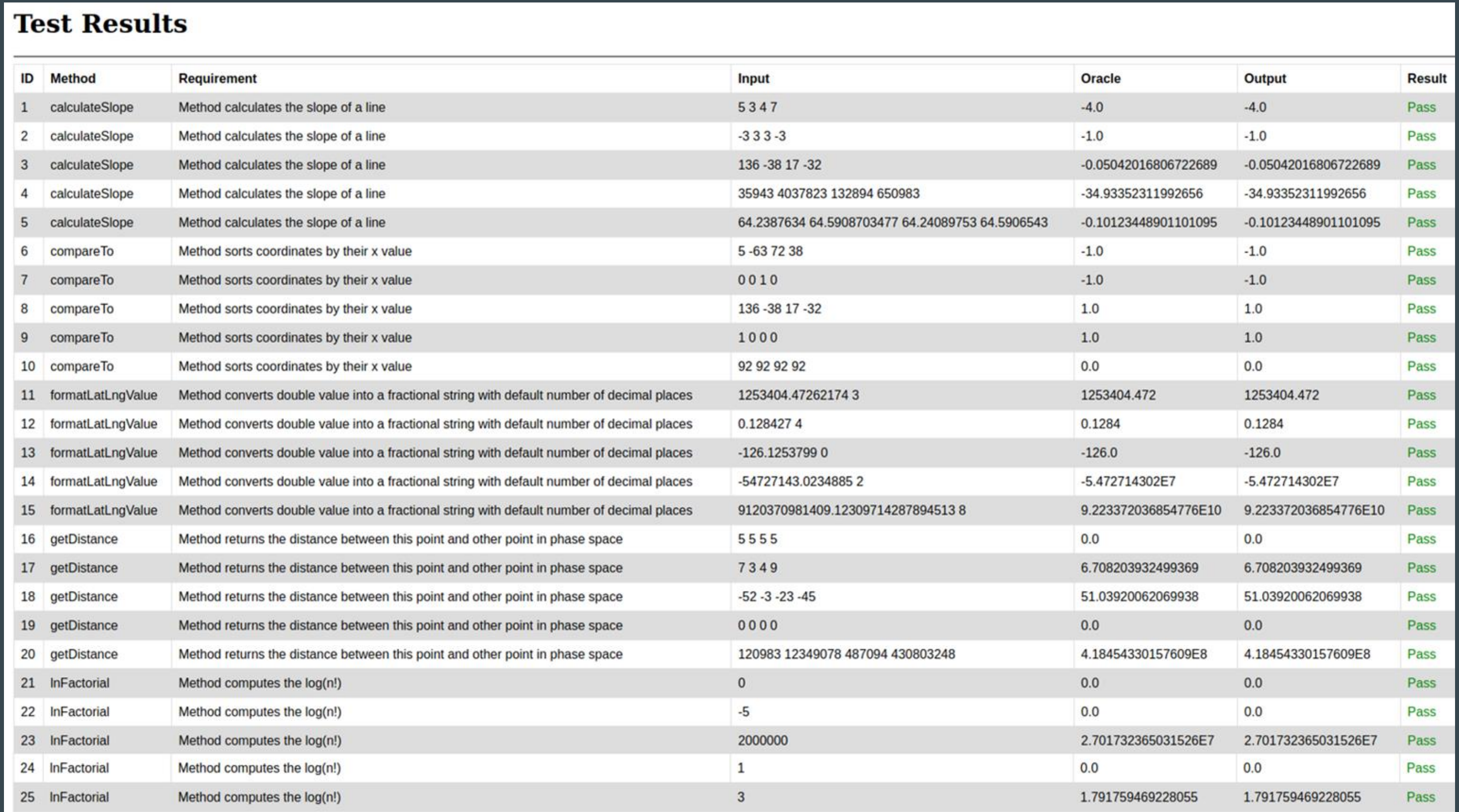
ID	Calculation	Input	Oracle	Output	Result
1	Compare points (5.000000,-63.000000) and (72.000000,38.000000)	5 -63 72 38	-1.0	-1.0	Pass
2	Compare points (0.000000,0.000000) and (1.000000,0.000000)	0 0 1 0	-1.0	-1.0	Pass
3	Compare points (136.000000,-38.000000) and (17.000000,-32.000000)	136 -38 17 -32	1.0	1.0	Pass
4	Compare points (1.000000,0.000000) and (0.000000,0.000000)	1 0 0 0	1.0	1.0	Pass
5	Compare points (92.000000,92.000000) and (92.000000,92.000000)	92 92 92 92	0.0	0.0	Pass

formatLatLngValue()

Method converts double value into a fractional string with default number of decimal places

ID	Calculation	Input	Oracle	Output	Result
1	Format 1253404.47262174	1253404.47262174 3	1253404.472	1253404.472	Pass
2	Format 0.128427	0.128427 4	0.1284	0.1284	Pass
3	Format -126.1253799	-126.1253799 0	-126.0	-126.0	Pass
4	Format -5.47271430234885E7	-54727143.0234885 2	-5.472714302E7	-5.472714302E7	Pass
5	Format 9.120370981409123E12	9120370981409.12309714287894513 8	9.223372036854776E10	9.223372036854776E10	Pass

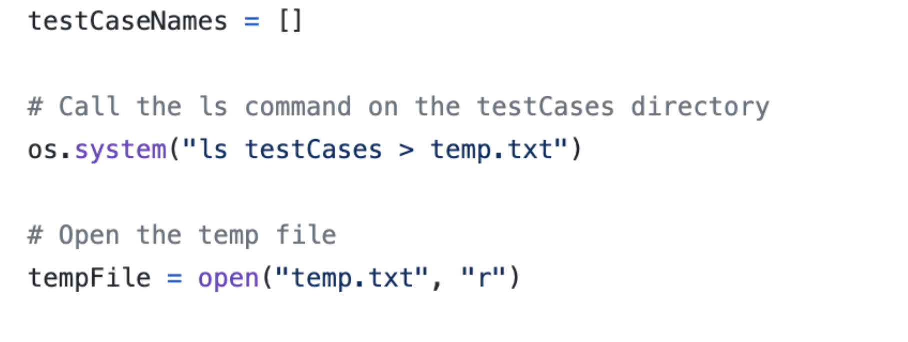
Final Test Results Report



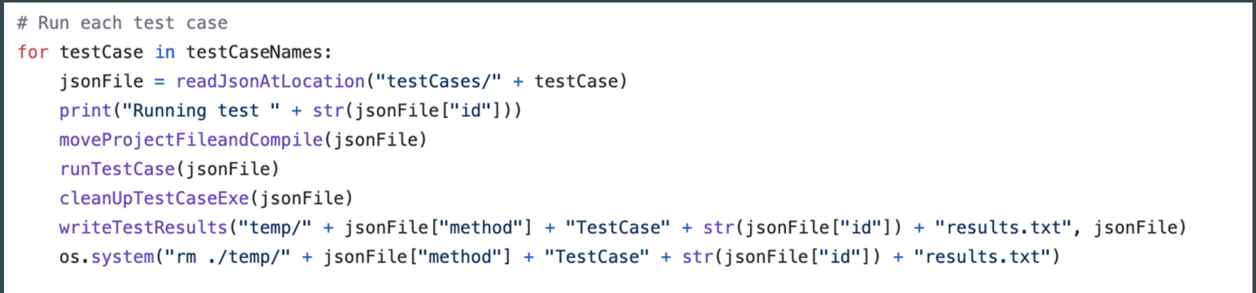
Test Results

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

Final Script Implementation (Main)

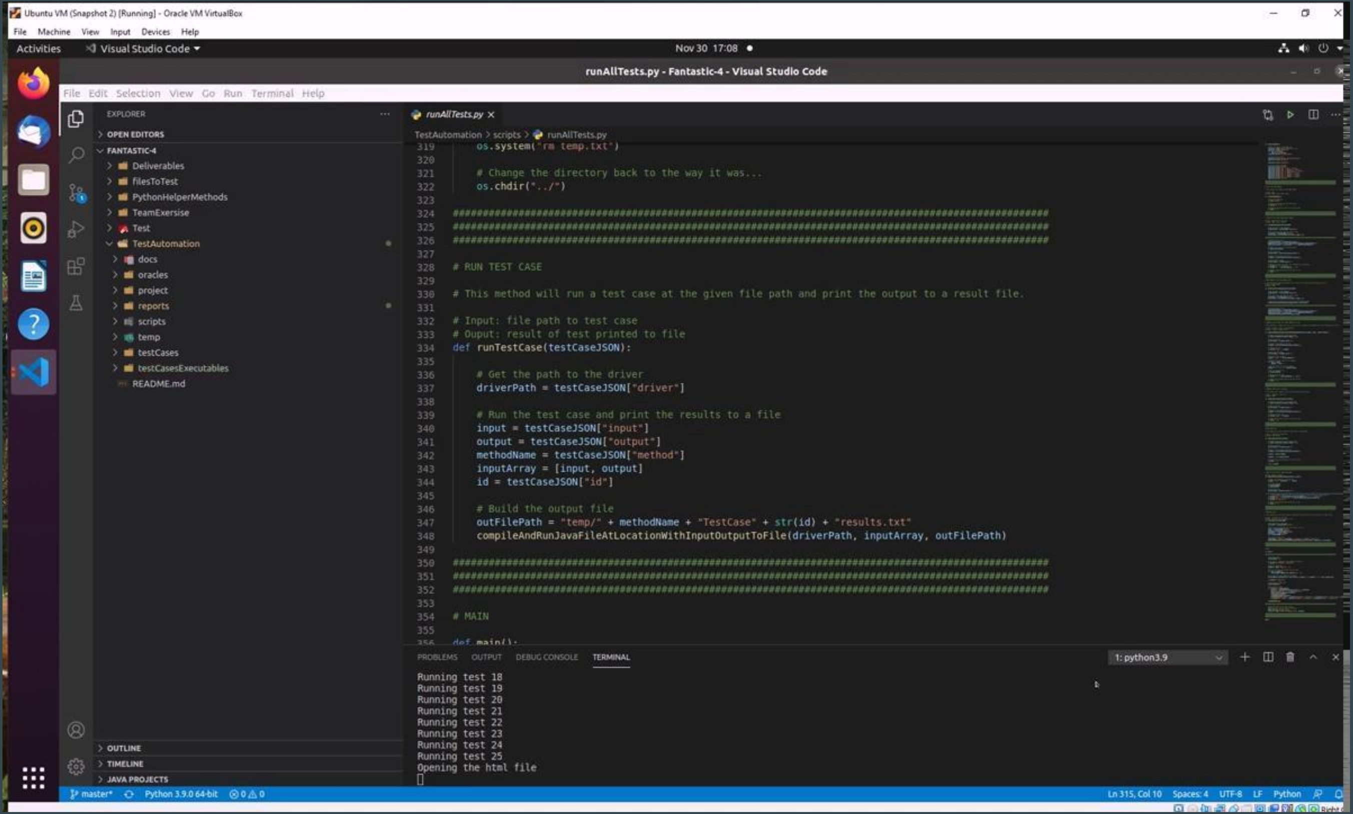


```python
testCaseNames = []

# Call the ls command on the testCases directory
os.system("ls testCases > temp.txt")

# Open the temp file
tempFile = open("temp.txt", "r")
```



```python
# Run each test case
for testCase in testCaseNames:
    jsonFile = readJsonAtLocation("testCases/" + testCase)
    print("Running test " + str(jsonFile["id"]))
    moveProjectFileandCompile(jsonFile)
    runTestCase(jsonFile)
    cleanUpTestCaseExe(jsonFile)
    writeTestResults("temp/" + jsonFile["method"] + "TestCase" + str(jsonFile["id"]) + "results.txt", jsonFile)
    os.system("rm ./temp/" + jsonFile["method"] + "TestCase" + str(jsonFile["id"]) + "results.txt")
```

Script Demo

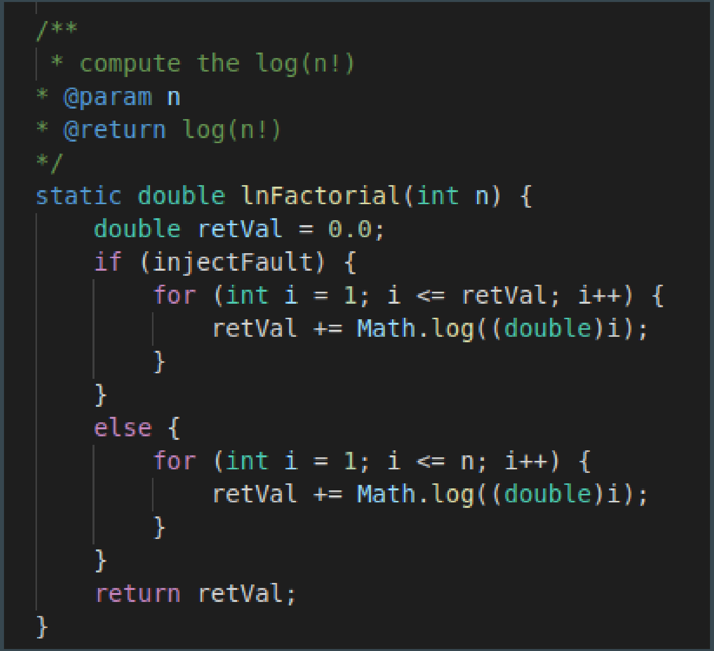


```
    os.system("rm temp.txt")

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

####################################################################################################
####################################################################################################
####################################################################################################

# RUN TEST CASE

# This method will run a test case at the given file path and print the output to a result file.

# Input: file path to test case
# Ouput: result of test printed to file
def runTestCase(testCaseJSON):

    # Get the path to the driver
    driverPath = testCaseJSON["driver"]

    # Run the test case and print the results to a file
    input = testCaseJSON["input"]
    output = testCaseJSON["output"]
    methodName = testCaseJSON["method"]
    inputArray = [input, output]
    id = testCaseJSON["id"]

    # Build the output file
    outFilePath = "temp/" + methodName + "TestCase" + str(id) + "results.txt"
    compileAndRunJavaFileAtLocationWithInputOutputToFile(driverPath, inputArray, outFilePath)

####################################################################################################
####################################################################################################
####################################################################################################

# MAIN

def main():
```

```
Running test 18
Running test 19
Running test 20
Running test 21
Running test 22
Running test 23
Running test 24
Running test 25
Opening the html file
```

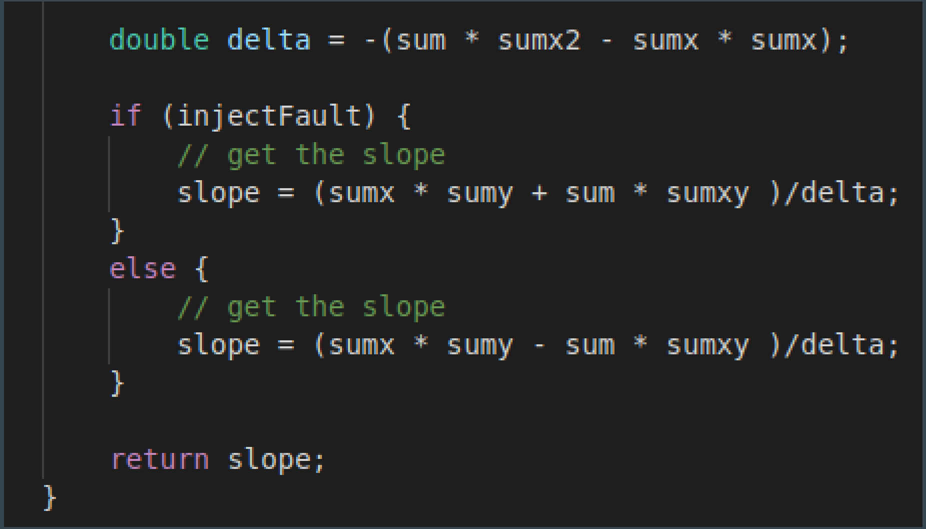
Faults

lnFactorial()



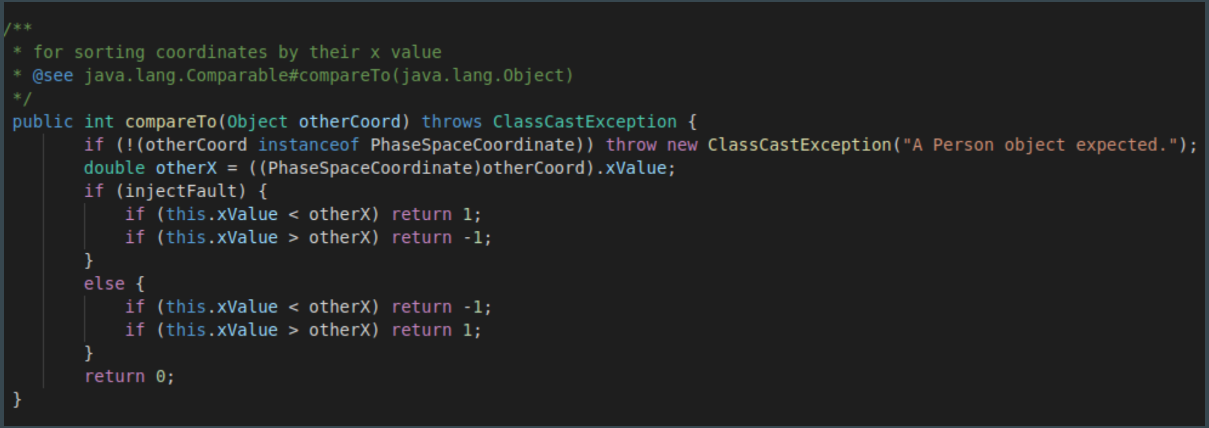
```java
/**
 * compute the log(n!)
* @param n
* @return log(n!)
*/
static double lnFactorial(int n) {
    double retVal = 0.0;
    if (injectFault) {
        for (int i = 1; i <= retVal; i++) {
            retVal += Math.log((double)i);
        }
    }
    else {
        for (int i = 1; i <= n; i++) {
            retVal += Math.log((double)i);
        }
    }
    return retVal;
}
```

calculateSlope()



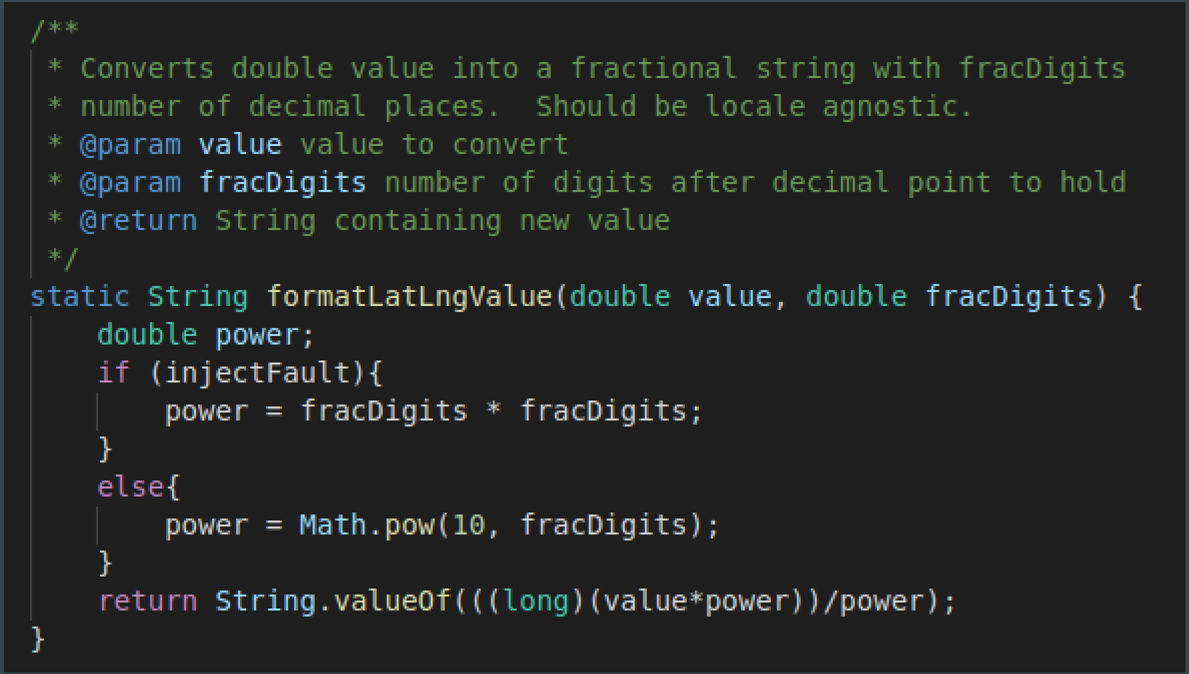
```
    double delta = -(sum * sumx2 - sumx * sumx);

    if (injectFault) {
        // get the slope
        slope = (sumx * sumy + sum * sumxy )/delta;
    }
    else {
        // get the slope
        slope = (sumx * sumy - sum * sumxy )/delta;
    }

    return slope;
}
```

compareTo()



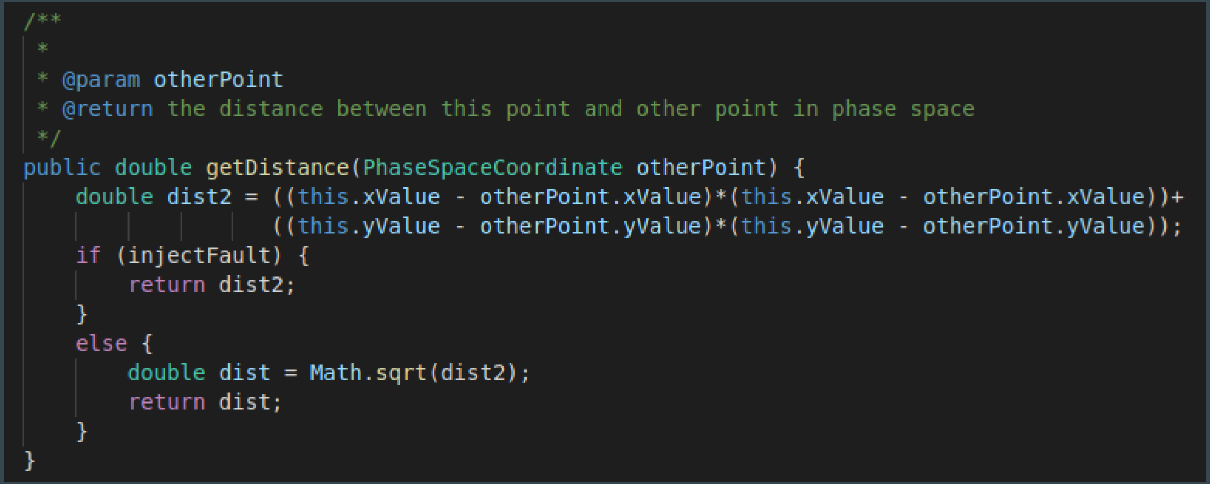
```java
/**
 * for sorting coordinates by their x value
 * @see java.lang.Comparable#compareTo(java.lang.Object)
 */
public int compareTo(Object otherCoord) throws ClassCastException {
        if (!(otherCoord instanceof PhaseSpaceCoordinate)) throw new ClassCastException("A Person object expected.");
        double otherX = ((PhaseSpaceCoordinate)otherCoord).xValue;
        if (injectFault) {
            if (this.xValue < otherX) return 1;
            if (this.xValue > otherX) return -1;
        }
        else {
            if (this.xValue < otherX) return -1;
            if (this.xValue > otherX) return 1;
        }
        return 0;
}
```

formatLatLngValue()



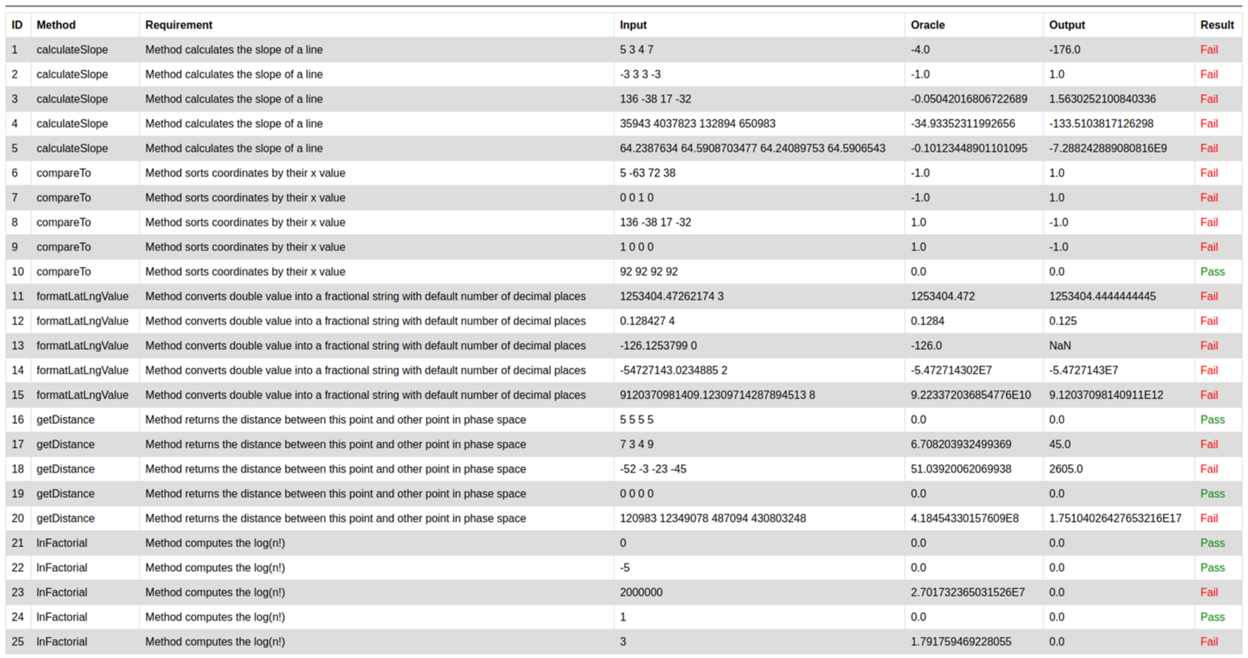
```java
/**
 * Converts double value into a fractional string with fracDigits
 * number of decimal places.  Should be locale agnostic.
 * @param value value to convert
 * @param fracDigits number of digits after decimal point to hold
 * @return String containing new value
 */
static String formatLatLngValue(double value, double fracDigits) {
    double power;
    if (injectFault){
        power = fracDigits * fracDigits;
    }
    else{
        power = Math.pow(10, fracDigits);
    }
    return String.valueOf(((long)(value*power))/power);
}
```

getDistance()



```
/**
 *
 * @param otherPoint
 * @return the distance between this point and other point in phase space
 */
public double getDistance(PhaseSpaceCoordinate otherPoint) {
    double dist2 = ((this.xValue - otherPoint.xValue)*(this.xValue - otherPoint.xValue))+
                   ((this.yValue - otherPoint.yValue)*(this.yValue - otherPoint.yValue));
    if (injectFault) {
        return dist2;
    }
    else {
        double dist = Math.sqrt(dist2);
        return dist;
    }
}
```

Test Result Report with Faults Live



ID	Method	Requirement	Input	Oracle	Output	Result
1	calculateSlope	Method calculates the slope of a line	5 3 4 7	-4.0	-176.0	Fail
2	calculateSlope	Method calculates the slope of a line	-3 3 3 -3	-1.0	1.0	Fail
3	calculateSlope	Method calculates the slope of a line	136 -38 17 -32	-0.05042016806722689	1.5630252100840336	Fail
4	calculateSlope	Method calculates the slope of a line	35943 4037823 132894 650983	-34.93352311992656	-133.5103817126298	Fail
5	calculateSlope	Method calculates the slope of a line	64.2387634 64.5908703477 64.24089753 64.5906543	-0.10123448901101095	-7.288242889080816E9	Fail
6	compareTo	Method sorts coordinates by their x value	5 -63 72 38	-1.0	1.0	Fail
7	compareTo	Method sorts coordinates by their x value	0 0 1 0	-1.0	1.0	Fail
8	compareTo	Method sorts coordinates by their x value	136 -38 17 -32	1.0	-1.0	Fail
9	compareTo	Method sorts coordinates by their x value	1 0 0 0	1.0	-1.0	Fail
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	1253404.4444444445	Fail
12	formatLatLngValue	Method converts double value into a fractional string with default number of decimal places	0.128427 4	0.1284	0.125	Fail
13	formatLatLngValue	Method converts double value into a fractional string with default number of decimal places	-126.1253799 0	-126.0	NaN	Fail
14	formatLatLngValue	Method converts double value into a fractional string with default number of decimal places	-54727143.0234885 2	-5.472714302E7	-5.4727143E7	Fail
15	formatLatLngValue	Method converts double value into a fractional string with default number of decimal places	9120370981409.12309714287894513 8	9.223372036854776E10	9.12037098140911E12	Fail
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	45.0	Fail
18	getDistance	Method returns the distance between this point and other point in phase space	-52 -3 -23 -45	51.03920062069938	2605.0	Fail
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	1.75104026427653216E17	Fail
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	0.0	Fail
24	lnFactorial	Method computes the log(n!)	1	0.0	0.0	Pass
25	lnFactorial	Method computes the log(n!)	3	1.791759469228055	0.0	Fail

Reflections on Fault Injection

Closing Thoughts

Questions? Comments?