Chapter 3:
Developing the Automated Testing Framework

Architectural Description

An extensive architectural description of the automated testing framework can be found on our repo via the path:

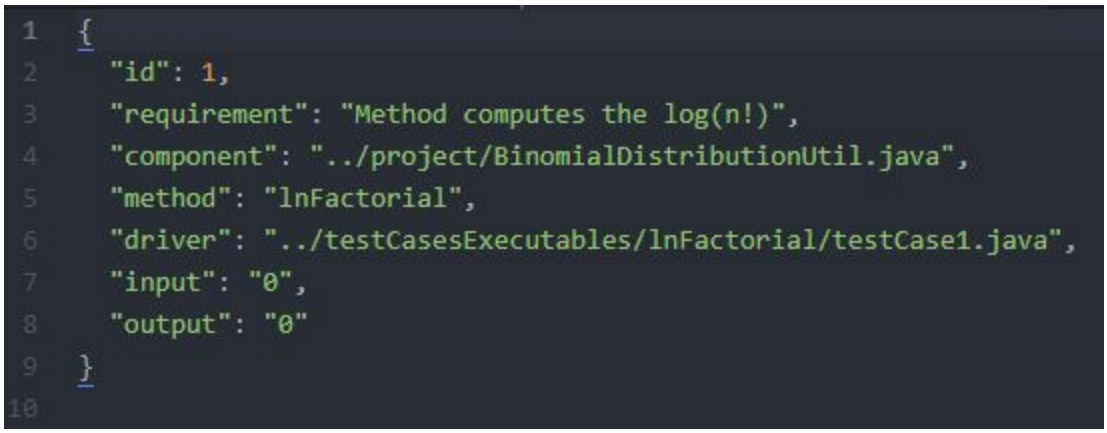
Deliverables/Deliverable 3/Architectural Description.pdf

How-To

1. Install Oracle VM VirtualBox.
    https://www.virtualbox.org/wiki/Downloads
2. Download a Linux ISO file.
3. Import the ISO file into Virtual Box.
4. Start the Linux VM.
5. Install the operating system.
6. Open a browser and navigate to the Fantastic 4 repo page.
7. Save the repository as a zip file.
8. Unzip the repo to a location on the computer.
9. Open a terminal.
10. Type the following commands…
    a. sudo apt-get update
    b. sudo apt-get install python3.9
    c. sudo apt-get install openjdk-8-jdk
11. Navigate to the folder that houses the downloaded repo.
12. Type the following commands…
    a. cd TestAutomation/scripts
    b. python3.9 runAllTests.py

Test Cases (lnFactorial)

In the previous chapter we specified 5 test cases for the method lnFactorial, which is used in the calculation of binomial distributions within the StochasticPoissonSIDiseaseModelImpl class.



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

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

runAllTests.py Script

Important methods from the runAllTests Script

runAllTests.py (runTestCase);

Takes a test case JSON and outputs the results to a file

```
# 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"]
    inputArray = [input, output]
    outFilePath = testCaseJSON["result"]
    compileAndRunJavaFileAtLocationWithInputOutputToFile(driverPath, inputArray, outFilePath)
```

runAllTests.py (testMethod);

```
def testMethod(methodName):
    pathToJSON = "../testCases/" + methodName + "/"
    testOne = readJsonAtLocation(pathToJSON + "testCase1.json")
    testTwo = readJsonAtLocation(pathToJSON + "testCase2.json")
    testThree = readJsonAtLocation(pathToJSON + "testCase3.json")
    testFour = readJsonAtLocation(pathToJSON + "testCase4.json")
    testFive = readJsonAtLocation(pathToJSON + "testCase5.json")

    # You only run this once per method...
    moveProjectFileandCompile(testOne)

    # You only run this once per method...
    cleanOutTempFoder(testOne)

    print("Testing " + methodName + ":")

    # Test case 1
    print("Running test 1")
    runTestCase(testOne)
    # Test case 2
    print("Running test 2")
    runTestCase(testTwo)
    # Test case 3
    print("Running test 3")
    runTestCase(testThree)
    # Test case 4
    print("Running test 4")
    runTestCase(testFour)
    # Test case 5
    print("Running test 5\n")
    runTestCase(testFive)

    # You only run this once per method...
    cleanUpTestCaseExe(testOne)
```

runAllTests.py (constructReport);

Constructs a report detailing the test cases and the results of each test case execution

```
def constructReport(methodNames):
    print("Constructing final report")

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

    for method in methodNames:
        writeMethodResults(method)
```