Automated Testing Framework for Spatiotemporal Epidemiological Modeler (STEM)

Ashanti Long, Clare Clever, Jim Bowring, Katherine Sweeney
College of Charleston

Introduction

In this project we constructed an automated testing framework for the STEM application. The script we built for this project runs in a Linux environment and executes 25 test cases.

Materials and methods

The items used for this project were GitHub, Java, Linux, Python and the STEM code base.

The script runs via the command-line interface (CLI) and utilizes bash commands to test each method.

```
Fantastic-4/TestAutomation$ python3.9 ./scripts/runAllTests.py
```

CLI command

```
import os
os.chdir("temp/")
os.system("rm -r *")
os.chdir("../")
```

Bash commands

Results

The STEM engine manipulates data taken as input to produce spatiotemporal models. To do this the code utilizes a plethora of mathematical functions. These functions are needed to compute related statistical values for modeling purposes. Most of the methods that we were interested in testing were methods related to mathematical calculations. The five methods we tested were: lnFactorial, getDistance, compareTo, calculateSlope, and formatLatLngValue. We created 5 tests for each method for a total of 25 test cases. The resulting output is shown below…

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	9.2233720368547766E10	9.2233720368547766E10	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	2.7017323650315 26E7	2.7017323650315 26E7	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

Conclusion

Working on this project was an incredible experience. We found that it was crucial to fully test each method. Faults can exist that are not going to cause errors to be thrown but could cause the method to behave in a way that it is not supposed to. This project demonstrated the importance of double-checking things. Additionally, we learned how to implement test cases to catch potential logic errors. Subsequently shining a light on the idea that errors are not going to be caught by the compiler 100% of the time. Logic errors need to be found via exhaustive testing, or through additional human scrutinization of the code.

Acknowledgments

We would love to thank Jim Bowring for his immense help throughout the duration of this project.

Further information

https://github.com/csci-362-02-2020/Fantastic-4