Names: Madison Betz and Kaiden Pollesch

Date: 1/31/2025

Assignment Title: Lab 1 Triangle Trouble

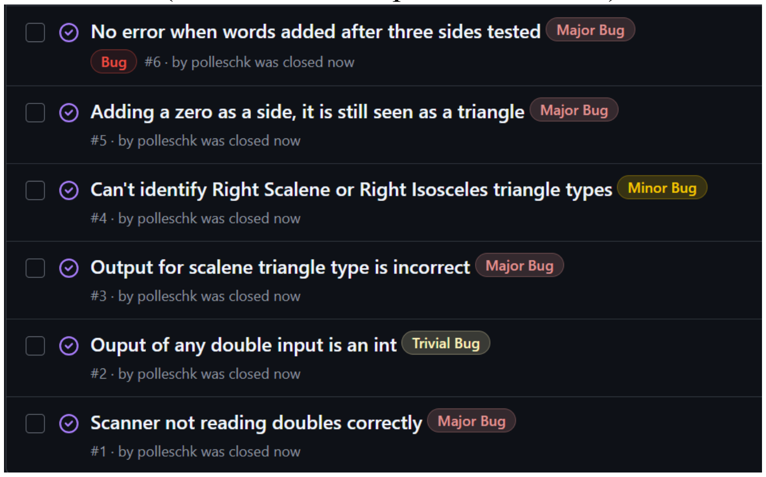
Repository Link: https://github.com/msoe-SWE2721/swe2721-lab1-2025-121_betz_pollesch_lab1_2025.git

Introduction

In this lab, we are working with a partner on existing software to test and modify it. The code is supposed to verify the validity of a triangle based on the mathematical rules that determine a triangle, however there are several bugs that cause the code to function differently. We are making tests with inputs and expected outputs along with generating reports to keep track of failures. After finding and analyzing the failures, we are updating the software to function correctly and managing our bug reports to reflect that.

Test Developer Discussion

We determined our test cases by creating many possible valid tests for the different triangle types and of various sizes. In addition to this, we also made sure to test possibilities that would not result in valid triangles to make sure the program handled correctly. To organize our test cases, we did it by the type of test based on triangle type, size, or other factors and then by application used to generate tests. We tried using Copilot and ChatGPT to generate some tests for cases we may not have thought to include. We found a total of six bugs in our code to fix with the screenshot from GitHub below (able to see descriptions in GitHub):



No error when words added after three sides tested Major Bug
Bug #6 · by polleschk was closed now

Adding a zero as a side, it is still seen as a triangle Major Bug
#5 · by polleschk was closed now

Can't identify Right Scalene or Right Isosceles triangle types Minor Bug
#4 · by polleschk was closed now

Output for scalene triangle type is incorrect Major Bug
#3 · by polleschk was closed now

Ouput of any double input is an int Trivial Bug
#2 · by polleschk was closed now

Scanner not reading doubles correctly Major Bug
#1 · by polleschk was closed now

Test Cases

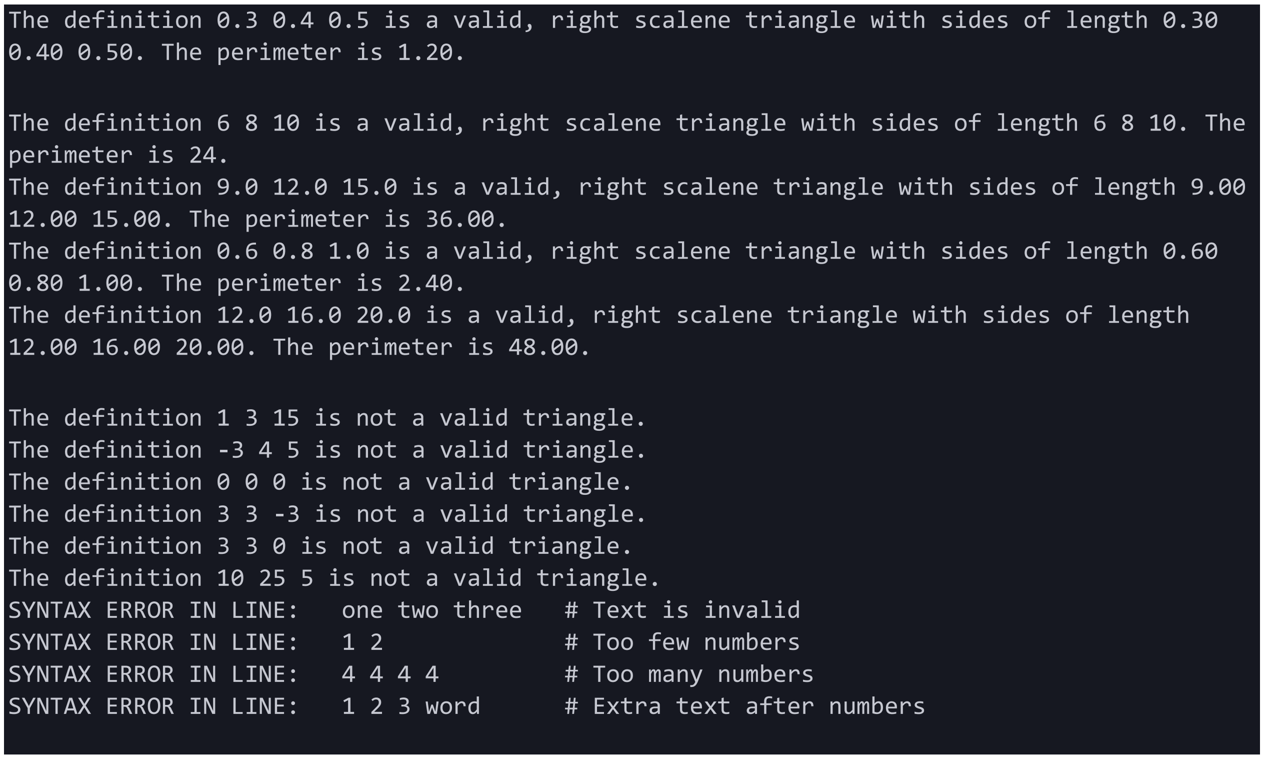
Test inputs:

```
3 3 3           # Basic equilateral triangle
5.0 5.0 5.0     # Floating-point equilateral triangle
100 100 100     # Large equilateral triangle
0.1 0.1 0.1     # Small equilateral triangle

1 3 3           # Basic isosceles triangle
2.5 2.5 4.0     # Floating-point isosceles triangle
3.0 4.0 4.0     # Another floating-point isosceles triangle
1000 1000 800   # Large isosceles triangle
0.5 0.5 0.3     # Small isosceles triangle

3 4 5           # Basic right scalene triangle
5 6 7           # General scalene triangle
6.0 8.0 10.0    # Floating-point right scalene triangle
```

```
7.1 9.2 10.3    # Arbitrary floating-point scalene triangle
100 150 200     # Large scalene triangle
0.3 0.4 0.5     # Small scalene triangle

6 8 10          # Right triangle (integer)
9.0 12.0 15.0   # Right triangle (floating-point)
0.6 0.8 1.0     # Small right triangle
12.0 16.0 20.0  # Floating-point right triangle

1 3 15          # Fails triangle inequality
-3 4 5          # Negative side length
0 0 0           # All sides zero
3 3 -3          # One negative side
3 3 0           # One side is zero
10 25 5         # Violates triangle inequality
one two three   # Text is invalid
1 2             # Too few numbers
4 4 4 4         # Too many numbers
1 2 3 word      # Extra text after numbers
```

Expected results:

```
The definition 3 3 3 is a valid, equilateral triangle with sides of length 3 3 3. The
perimeter is 9.
The definition 5.0 5.0 5.0 is a valid, equilateral triangle with sides of length 5.00
5.00 5.00. The perimeter is 15.00.
The definition 100 100 100 is a valid, equilateral triangle with sides of length 100 100
100. The perimeter is 300.
The definition 0.1 0.1 0.1 is a valid, equilateral triangle with sides of length 0.10
0.10 0.10. The perimeter is 0.30.

The definition 1 3 3 is a valid, isosceles triangle with sides of length 1 3 3. The
perimeter is 7.
The definition 2.5 2.5 4.0 is a valid, isosceles triangle with sides of length 2.50 2.50
4.00. The perimeter is 9.00.
The definition 3.0 4.0 4.0 is a valid, isosceles triangle with sides of length 3.00 4.00
4.00. The perimeter is 11.00.
The definition 1000 1000 800 is a valid, isosceles triangle with sides of length 1000
1000 800. The perimeter is 2800.
The definition 0.5 0.5 0.3 is a valid, isosceles triangle with sides of length 0.50 0.50
0.30. The perimeter is 1.30.

The definition 3 4 5 is a valid, right scalene triangle with sides of length 3 4 5. The
perimeter is 12.
The definition 5 6 7 is a valid, scalene triangle with sides of length 5 6 7. The
perimeter is 18.
The definition 6.0 8.0 10.0 is a valid, right scalene triangle with sides of length 6.00
8.00 10.00. The perimeter is 24.00.
The definition 7.1 9.2 10.3 is a valid, scalene triangle with sides of length 7.10 9.20
10.30. The perimeter is 26.60.
The definition 100 150 200 is a valid, scalene triangle with sides of length 100 150 200.
The perimeter is 450.
```



```
The definition 0.3 0.4 0.5 is a valid, right scalene triangle with sides of length 0.30
0.40 0.50. The perimeter is 1.20.

The definition 6 8 10 is a valid, right scalene triangle with sides of length 6 8 10. The
perimeter is 24.
The definition 9.0 12.0 15.0 is a valid, right scalene triangle with sides of length 9.00
12.00 15.00. The perimeter is 36.00.
The definition 0.6 0.8 1.0 is a valid, right scalene triangle with sides of length 0.60
0.80 1.00. The perimeter is 2.40.
The definition 12.0 16.0 20.0 is a valid, right scalene triangle with sides of length
12.00 16.00 20.00. The perimeter is 48.00.

The definition 1 3 15 is not a valid triangle.
The definition -3 4 5 is not a valid triangle.
The definition 0 0 0 is not a valid triangle.
The definition 3 3 -3 is not a valid triangle.
The definition 3 3 0 is not a valid triangle.
The definition 10 25 5 is not a valid triangle.
SYNTAX ERROR IN LINE:   one two three   # Text is invalid
SYNTAX ERROR IN LINE:   1 2             # Too few numbers
SYNTAX ERROR IN LINE:   4 4 4 4         # Too many numbers
SYNTAX ERROR IN LINE:   1 2 3 word      # Extra text after numbers
```

Developer / Bug Fixer

In total, we found six bugs: missing error when extra words were added after the number inputs, missing error when zero was set as a side, right triangles were not correctly printed, output for scalene triangle was not printing correctly, and the output of any input with a double variable type would result in an integer output. The descriptions within the GitHub issue reports by the tester made understanding the root problem and fixing the bugs much easier. They directly stated what the issue was and where/how it should be fixed in the code.

Things gone right / Things gone wrong

Items that went correctly during this exercise were using GitHub issue tracking, creating useful tests and their expected outputs, and finding and fixing the bugs within the code. Using the GitHub tracking allowed us to efficiently work together to take care of bugs that were present in the software by recording useful information to quickly find and fix bugs. Using a test that tested both correct inputs and incorrect inputs allowed us to make sure that the code would function as expected for various cases. We also did not have much trouble finding the bugs and fixing them because of how our tests and issues were written.
Something that did not go well for this lab was the set up. We were not able to get the code with Gradle to be properly set up on our computers for over an hour and had to completely re-clone the files several times, having issues with it recognizing packages and folders correctly.

Conclusions

What have you learned from this experience?

We learned that having limited tests only allows us to catch certain cases where code could be going wrong. We learned that having more expensive testing allows the developers and tester so understand how their code is working well, or how it is working not as expected. Having the edge case testing showed us exactly what is going wrong, so we didn’t have to sift through every line of code to fine where our errors are occurring. Although the errors might not crash the system it was important to make sure that everything worked as we expected, and everything was being outputted to the user as expected.

How has this lab experience changed your attitude toward testing and when you should work on testing your projects?

This lab allowed us to experience exactly how almost everything can go wrong, and when having very limited and simple tests it can look like the whole code is working as it should. It also showed that testing is super important to ensure that everything is working as expected. In a way there should be testing throughout the entire project so that small errors can be caught right away rather than later on, and when they’re caught early on it shouldn’t take as long to track down the problem and resolve the issue.