**Assignment Description:** Complete the TestTriangle test cases, fix logic in classifyTriangle function.

**Author:** Ejona Kocibelli

**Summary:**

The initial test case result performed in the original classifyTriangle function. I tested the categories: WRONG INPUTS, NOT A TRIANGLE, EQUILATERAL, ISOSCELES, SCALENE, RIGHT AND SCALENE, RIGHT AND ISOSCELES. For each of these categories I included different tests.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test  Category | Test ID | Input | Expected Result | Actual Result | Pass or Fail |
| WRONG  INPUTS | 01 | 'hello', 1, 1 | **InvalidInput** | **TypeError: '>' not supported between instances of 'str' and 'int'** | FAIL |
|  | 02 | 'A', 'B', 'C' | **InvalidInput** | **TypeError: '>' not supported between instances of 'str' and 'int'** | FAIL |
|  | 03 | -1, -3, -2 | **InvalidInput** | **InvalidInput** | PASS |
|  | 04 | 0, 0, 0 | **InvalidInput** | **InvalidInput** | PASS |
| NOT  TRIANGLE | 05 | 1, 1, 2 | **NotATriangle** | **InvalidInput** | FAIL |
|  | 06 | 7, 3, 2 | **NotATriangle** | **InvalidInput** | FAIL |
|  | 07 | 1, 2, 3 | **NotATriangle** | **InvalidInput** | FAIL |
| EQUILATERAL | 08 | 1, 1, 1 | **Equilateral** | **InvalidInput** | FAIL |
|  | 09 | 3.33, 3.33, 3.33 | **Equilateral** | **InvalidInput** | FAIL |
|  | 10 | 1e0, 1e0, 1e0 | **Equilateral** | **InvalidInput** | FAIL |
| ISOSCELES | 11 | 1, 3, 1 | **Isosceles** | **InvalidInput** | FAIL |
|  | 12 | 1234567890, 1234567890, 987654321 | **InvalidInput** | **InvalidInput** | PASS |
|  | 13 | 2, 2, 2.000000000000001 | **Isosceles** | **InvalidInput** | FAIL |
| SCALENE | 14 | 3, 4, 6 | **Scalene** | **InvalidInput** | FAIL |
|  | 15 | 4, 3, 4 | **Scalene** | **InvalidInput** | FAIL |
|  | 16 | 3 \* (2 ^ 64), 4 \* (2 ^ 64), 5 \* (2 ^ 64) | **InvalidInput** | **InvalidInput** | PASS |
| RIGHT AND SCALENE | 17 | 2, 3, 4 | **Right, Scalene** | **InvalidInput** | FAIL |
|  | 18 | 3, 4, 5 | **Right, Scalene** | **InvalidInput** | FAIL |
|  | 19 | 3, 5, 4 | **Right, Scalene** | **InvalidInput** | FAIL |
|  | 20 | 3000000, 4000000, 5000000 | **InvalidInput** | **InvalidInput** | PASS |
| RIGHT AND ISOSCELES | 21 | 1, 1, math.sqrt(2) | **Right, Isosceles** | **InvalidInput** | FAIL |
|  | 22 | 2147483647, 2147483647, 5 | **InvalidInput** | **InvalidInput** | PASS |
|  | 23 | 3, 3, 4.242640687119285146 | **Right, Isosceles** | **InvalidInput** | FAIL |

As shown in the table above, most of the test cases failed for the initial version of classifyTriangle. Thus, changes in the initial version are necessary. Below is a list of changes, I will perform to classifyTriangle function.

* Side of the triangle inputs should be only integers and the program should raise an Exception for any other input.
* Triangle cannot have negative and zero sides, the input is an integer, thus it is valid input but it is not a triangle so it should notify the user that triangles cannot have 0 and Negative sides.
* Sides of the Triangle should be compliant with Triangle Side Rule. The program should notify the user if the inputs enter do not form a triangle
* Triangles should be classified: Equilateral, Isosceles, Scalene, Right Scalene, Right Isosceles. A triangle cannot be only Right, it is whether Right Scalene or Right Isosceles.
* Mathematically, triangle sides should not be restricted up to 200, but I kept this limit for the assignment.
* Input sides of triangles should give the same result of the triangle classification regardless the order of the input

I implemented the changes above in the classifyTriangle functions and run again the test cases. The tests cases passed successfully.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test  Category | Test ID | Input | Expected Result | Actual Result | Pass or Fail |
| WRONG  INPUTS | 01 | 'hello', 1, 1 | **InvalidInput** | **InvalidInput** | PASS |
|  | 02 | 'A', 'B', 'C' | **InvalidInput** | **InvalidInput** | PASS |
|  | 03 | -1, -3, -2 | **InvalidInput** | **InvalidInput** | PASS |
|  | 04 | 0, 0, 0 | **InvalidInput** | **InvalidInput** | PASS |
| NOT  TRIANGLE | 05 | 1, 1, 2 | **NotATriangle** | **NotATriangle** | PASS |
|  | 06 | 7, 3, 2 | **NotATriangle** | **NotATriangle** | PASS |
|  | 07 | 1, 2, 3 | **NotATriangle** | **NotATriangle** | PASS |
| EQUILATERAL | 08 | 1, 1, 1 | **Equilateral** | **Equilateral** | PASS |
|  | 09 | 3.33, 3.33, 3.33 | **Equilateral** | **Equilateral** | PASS |
|  | 10 | 1e0, 1e0, 1e0 | **Equilateral** | **Equilateral** | PASS |
| ISOSCELES | 11 | 1, 3, 1 | **Equilateral** | **Equilateral** | PASS |
|  | 12 | 1234567890, 1234567890, 987654321 | **InvalidInput** | **InvalidInput** | PASS |
|  | 13 | 2, 2, 2.000000000000001 | **Isosceles** | **Isosceles** | PASS |
| SCALENE | 14 | 3, 4, 6 | **Scalene** | **Scalene** | PASS |
|  | 15 | 4, 3, 4 | **Scalene** | **Scalene** | PASS |
|  | 16 | 3 \* (2 ^ 64), 4 \* (2 ^ 64), 5 \* (2 ^ 64) | **InvalidInput** | **InvalidInput** | PASS |
| RIGHT AND SCALENE | 17 | 2, 3, 4 | **Right and Scalene** | **Right and Scalene** | PASS |
|  | 18 | 3, 4, 5 | **Right and Scalene** | **Right and Scalene** | PASS |
|  | 19 | 3, 5, 4 | **Right and Scalene** | **Right and Scalene** | PASS |
|  | 20 | 3000000, 4000000, 5000000 | **InvalidInput** | **InvalidInput** | PASS |
| RIGHT AND ISOSCELES | 21 | 1, 1, math.sqrt(2) | **Right and Isosceles** | **Right and Isosceles** | PASS |
|  | 22 | 2147483647, 2147483647, 5 | **InvalidInput** | **InvalidInput** | PASS |
|  | 23 | 3, 3, 4.242640687119285146 | **Right and Isosceles** | **Right and Isosceles** | PASS |

Summary Result

|  |  |  |
| --- | --- | --- |
|  | Test Run 1 | Test Run 2 |
| Tests Planned | 23 | 23 |
| Tests Executed | 23 | 23 |
| Tests Passed | 6 | 23 |
| Defects Found | 17 | 0 |
| Defects Fixed | 17 | 0 |

**Reflection:** In this assignment, I learnt how to implement test cases before you start coding. I prepared my test cases, before I started to improve the original file. I spend time on focusing on the errors and failures, and finding ways how to approach to solve them, and I managed the program to successfully run all the test cases in one attempt.

**Honor pledge:** I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination. I further pledge that I have not copied any material from a book, article, the Internet or any other source except where I have expressly cited the source.

**Detailed Results:** Attached I will post screenshots of my test results. The constraints I used in the code, were the Triangle Side Rule, Triangle Classification, Integer input and the the range of inputs should be bigger than zero and up to 200. Data I tested was integers, floats, characters, strings, negative, positive, high numbers, small numbers etc.

A screenshot of a computer

Description automatically generated