CMPE 187

MUTATION TESTING

GROUP #7

Table of Contents

[**I) Problem Description**](#_bd6mnrys48hx) **3**

[**II) Program Description**](#_ie2ifx1garcm) **3**

[**III) Program Source Code**](#_zhqfieas5dh0) **4**

[**IV) Test Oracle & Test Suite**](#_t0njltya7tj5) **6**

[Environment & OS](#_gf8cpdioqusc) 6

[Strategy for Testing](#_4i03r75gxi6c) 6

[Test Suite](#_zihvch469ski) 6

[**V) Mutation Testing**](#_mo7eqmifikuh) **9**

# 

# 

# I) Problem Description

In this homework assignment, we will design an Area Calculator in Java. We will create test cases for this program and run them. After that, we will introduce mutations to the original source code and run the test cases again. Finally, we will calculate the mutation score and judge the test cases based on that.

# II) Program Description

This is a Java program that will calculate the area of a Triangle, a Rectangle, a Circle or a Square. Firstly, the user will be asked to choose which shape they want to calculate by entering in 1 for a Rectangle, 2 for a Square, 3 for a Triangle and 4 for a Circle. After that, if the user enters 1, then they need to input the length and width of a rectangle. The program will check these values and gives the user their answer. If the user enters 2, then they need to input the length of a square. For option 3, the user needs to input the base and height of the triangle in order for the program to calculate the result. Lastly, for option 4, the user needs to input the radius of a circle, the program will calculate the area based on the provided values. For calculating the area of all the above shapes, If the user enters a negative number or a letter, the program will throw an exception.

# 

# 

# 

# III) Program Source Code

* **package** HW2;
* /\*
* This program will calculate the area of a Rectangle, a Triangle, a Square or a Circle based on provided values from the user.
* \*/
* **import** java.util.Scanner;
* **public** **class** AreaCalculator {
* //method to ask for double inputs from the user
* **private** **static** **double** ask(Scanner scanner, String prompt) {
* System.out.print(prompt);
* **return** scanner.nextDouble();
* }
* //method to calculate the area of the rectangle
* **private** **static** **void** rectangle(Scanner scanner) {
* **double** length = ask(scanner, "Please enter the length of the rectangle: ");
* **double** width = ask(scanner, "Please enter the width of the rectangle: ");
* //if the length and width of the rectangle is <=0, then throws an exception
* //otherwise calculate the area
* **if** (length <= 0 || width <= 0) {
* **throw** **new** IllegalArgumentException("Length or width cannot be 0 or negative");
* } **else** {
* System.out.println("Area of the rectangle is: " + (length \* width));
* }
* }
* //method to calculate the area of the square
* **private** **static** **void** square(Scanner scanner) {
* //if the length of a side is <=0, then throws exception
* **double** lengthOneSide = ask(scanner, "Please enter the length of 1 side: ");
* **if** (lengthOneSide <= 0) {
* **throw** **new** IllegalArgumentException("Length of a side cannot be 0 or negative");
* } **else** {
* System.out.println(lengthOneSide \* lengthOneSide);
* }
* }
* //method to calculate the area of the triangle
* **private** **static** **void** triangle(Scanner scanner) {
* **double** Base = ask(scanner, "Please enter the base of the triangle: ");
* **double** Height = ask(scanner, "Please enter the height the triangle: ");
* //if the base and height of the triangle is <=0, then throws exception
* **if** (Base <= 0 || Height <= 0) {
* **throw** **new** IllegalArgumentException("Length or Base cannot be 0 or negative");
* } **else** {
* System.out.println(0.5 \* Base \* Height);
* }
* }
* //method to calculate the area of the circle
* **private** **static** **void** circle(Scanner scanner) {
* **double** radius = ask(scanner, "Please enter the radius of the circle: ");
* //if the radius is <=0, then terminates
* **if** (radius <= 0) {
* **throw** **new** IllegalArgumentException("Length of the radius cannot be 0 or negative");
* } **else** {
* System.out.println(Math.PI \* (radius \* radius));
* }
* }

* //driver to ask for which area the user wants to calculate
* **public** **static** **void** main(String[] args) {
* Scanner scanner = **new** Scanner(System.in);
* System.out.println(
* "Please enter 1 to find the area of a Rectangle:\n" +
* "Enter 2 to find the area of a Square:\n" +
* "Enter 3 to find the area of a Triangle:\n" +
* "Enter 4 to find the area of a Circle:");
* **switch** (scanner.nextInt()) {
* **case** 1:
* rectangle(scanner);
* **break**;
* **case** 2:
* square(scanner);
* **break**;
* **case** 3:
* triangle(scanner);
* **break**;
* **case** 4:
* circle(scanner);
* **break**;
* }
* }
* }

# 

# 

# 

# 

# 

# IV) Test Oracle & Test Suite

## Environment & OS

Sony Vaio Flip 15 (2015)

Windows 10

Intel Core i5 Dual Core

8GB RAM

## Strategy for Testing

1. Area of a Rectangle = Length \* Width
2. Area of a Square = Length \* Length
3. Area of a Triangle = 0.5 \* Base \* Height
4. Area of a Circle = \*radius

## Test Suite

Test Cases for Area of a **Rectangle**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case # | Input (L,W) | Reason for Testing | Expected Output | Actual Output | Pass or Fail |
| 1 | 1, 1 | Robustness, completeness | 1 | 1 | Pass |
| 2 | a, b | Robustness, testing with special input values | exception threw | exception threw | Pass |
| 3 | 0, 5 | Robustness, testing with special input values | Illegal exception threw | Illegal exception threw | Pass |
| 4 | 0, 0 | Robustness, testing with special input values | Illegal exception threw | Illegal exception threw | Pass |
| 5 | -3, 5 | Robustness, testing with for invalid input. | Illegal exception threw | Illegal exception threw | Pass |
| 6 | 99999, 99999 | Brute force testing, robustness | Area of the rectangle is: 9.99890001E8 | Area of the rectangle is: 9.99890001E8 | Pass |
| 7 | 10, 15 | test for area of rectangle | Area of the rectangle is: 150.0 | Area of the rectangle is: 150.0 | Pass |

Test Cases for area of a **Square:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case # | Input (L,L) | Reason for Testing | Expected Output | Actual Output | Pass or Fail |
| 15 | 1, 1 | Robustness, completeness | 1 | 1 | Pass |
| 16 | a, b | Robustness, testing with special input values | Threw exception | Threw exception | Pass |
| 17 | 0, 5 | Robustness, testing with special input values | Illegal exception threw | Illegal exception threw | Pass |
| 18 | 0, 0 | Robustness, testing with special input values | Illegal exception threw | Illegal exception threw | Pass |
| 19 | -3, 5 | Robustness, testing with for invalid input. | Illegal exception threw | Illegal exception threw | Pass |
| 20 | 99999, 99999 | Brute force testing, robustness | 9.999800001E9 | 9.999800001E9 | Pass |
| 21 | 10, 15 | test for area of square | 100.0 | 100.0 | Pass |

Test Cases for area of a **Triangle:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case # | Input (B,H) | Reason for Testing | Expected Output | Actual Output | Pass or Fail |
| 8 | 1, 1 | Robustness, completeness | 1 | 1 | Pass |
| 9 | a, b | Robustness, testing with special input values | Exception threw | Exception thew | Pass |
| 10 | 0, 5 | Robustness, testing with special input values | Illegal exception threw | Illegal exception threw | Pass |
| 11 | 0, 0 | Robustness, testing with special input values | Illegal exception threw | Illegal exception threw | Pass |
| 12 | -3, 5 | Robustness, testing with invalid input. | Illegal exception threw | Illegal exception thew | Pass |
| 13 | 99999, 99999 | Brute force testing, robustness | 499000.5 | 499000.5 | Pass |
| 14 | 10, 15 | test for area of triangle | 75.0 | 75.0 | Pass |

Test Cases for area of a **Circle:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case # | Input (R) | Reason for Testing | Expected Output | Actual Output | Pass or Fail |
| 22 | 1 | Robustness, completeness | 3.141592653589793 | 3.141592653589793 | Pass |
| 23 | a | Robustness, testing with special input values | Threw exception | Threw exception | Pass |
| 24 | 0 | Robustness, testing with special input values | Illegal exception threw | Illegal exception threw | Pass |
| 25 | -3 | Robustness, testing with for invalid input. | Illegal exception threw | Illegal exception threw | Pass |
| 26 | 99999 | Brute force testing, robustness | 3.1415298220508804E10 | 3.1415298220508804E10 | Pass |
| 27 | 10 | test for area of circle | 314.1592653589793 | 314.1592653589793 | Pass |

# V) Mutation Testing

**Mutation 1:**

Line 22: System.out.println("Area of the rectangle is: " + (length \* width));

Change to: System.out.println("Area of the rectangle is: " + (length \* length));

**Mutation 2:**

Line 32: System.out.println(lengthOneSide \* lengthOneSide);

Change to: System.out.println(Math.pow(lengthOneSide, 2));

**Mutation 3:**

Line 43: System.out.println(0.5 \* Base \* Height);

Change to: System.out.println(0.5 \* Base \* Base);

**Mutation 4:**

Line 53: System.out.println(Math.PI \* (radius \* radius));

Change to: System.out.println(Math.PI \* (Math.pow(radius, 2)));

**Mutation 5:**

Line 15: if (length <= 0 || width <= 0)

Change to: if (length >= 0 || width == 0)

**Mutation 6:**

Line 24: if (lengthOneSide <= 0)

Change to: if (lengthOneSide > 0)

**Mutation 7:**

Line 34: if (Base <= 0 || Height < 0)

Change to: if (Base >= 0 || Height >= 0)

**Mutation 8:**

Line 43: if (radius <= 0)

Change to: if (radius >= 0)

**VI) Results of Mutation Testing**

**Mutation 1:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case # | Input | Expected Output | Actual Output | Pass/Fail |
| 1 | 1,1 | 1 | 1 | Pass |
| 2 | a,b | exception threw | exception threw | Pass |
| 3 | 0,5 | exception threw | exception threw | Pass |
| 4 | -3,5 | exception threw | exception threw | Pass |
| 5 | 10,15 | Area of the rectangle is: 150.0 | Area of the rectangle is: 100.0 | Fail |

**Mutation 2:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case # | Input | Expected Output | Actual Output | Pass/Fail |
| 1 | 1,1 | 1 | 1 | Pass |
| 2 | a,b | Threw exception | Threw exception | Pass |
| 3 | 0,5 | Threw exception | Threw exception | Pass |
| 4 | -3,5 | Illegal exception threw | Illegal exception threw | Pass |
| 5 | 10,15 | 100.0 | 100.0 | Pass |

**Mutation 3:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case # | Input | Expected Output | Actual Output | Pass/Fail |
| 1 | 1,1 | 0.5 | 0.5 | Pass |
| 2 | a,b | Threw exception | Threw exception | Pass |
| 3 | 0,5 | Threw exception | Threw exception | Pass |
| 4 | -3,5 | Illegal exception threw | Illegal exception threw | Pass |
| 5 | 10,15 | 75.0 | 50.0 | Fail |

**Mutation 4:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case # | Input | Expected Output | Actual Output | Pass/Fail |
| 1 | 1 | 3.141592653589793 | 314.1592653589793 | Pass |
| 2 | a | Exception threw | Exception threw | Pass |
| 3 | 0 | Exception threw | Exception threw | Pass |
| 4 | -3 | Illegal exception threw | Illegal exception threw | Pass |
| 5 | 10 | 314.1592653589793 | 314.1592653589793 | Pass |

**Mutation 5:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case # | Input | Expected Output | Actual Output | Pass/Fail |
| 1 | 5,7 | 35 | Length or width cannot be 0 or negative | Fail |
| 2 | a,b | Illegal exception threw | Illegal exception threw | Pass |
| 3 | 0,5 | Length or width cannot be 0 or negative | Length or width cannot be 0 or negative | Pass |
| 4 | -3,10 | Length or width cannot be 0 or negative | -30 | Fail |
| 5 | 10,18 | 180 | Length or width cannot be 0 or negative | Fail |

**Mutation 6:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case # | Input | Expected Output | Actual Output | Pass/Fail |
| 1 | 8 | 64.0 | Length of a side cannot be 0 or negative | Fail |
| 2 | a | Illegal exception threw | Illegal exception threw | Pass |
| 3 | 0 | Length of a side cannot be 0 or negative | 0.0 | Fail |
| 4 | -5 | Length of a side cannot be 0 or negative | -25 | Fail |
| 5 | 10 | 100 | Length of a side cannot be 0 or negative | Fail |

**Mutation 7:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case # | Input | Expected Output | Actual Output | Pass/Fail |
| 1 | 5,7 | 17.5 | Length or Base cannot be 0 or negative | Fail |
| 2 | a,b | Illegal exception threw | Illegal exception threw | Pass |
| 3 | 0,5 | Length or Base cannot be 0 or negative | Length or Base cannot be 0 or negative | Pass |
| 4 | -3,10 | Length or Base cannot be 0 or negative | Length or Base cannot be 0 or negative | Pass |
| 5 | 10,18 | 90 | Length or Base cannot be 0 or negative | Fail |

**Mutation 8:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case # | Input | Expected Output | Actual Output | Pass/Fail |
| 1 | 8 | 201.06 | Length of the radius cannot be 0 or negative | Fail |
| 2 | a | Illegal exception threw | Illegal exception threw | Pass |
| 3 | 0 | Length of the radius cannot be 0 or negative | Length of the radius cannot be 0 or negative | Pass |
| 4 | -5 | Length of the radius cannot be 0 or negative | 78.53 | Fail |
| 5 | 10 | 314.15 | Length of the radius cannot be 0 or negative | Fail |

After going through all the mutations with test cases, we see that the test cases are able to find errors at all 8 mutations, which meant the test cases killed all of the mutants.

Mutation score:

With N=8, D=8, E=0. Therefore,

**VII) Conclusion**

From the above mutation score, we can conclude that our test cases are acceptable in testing the program.

# 

# 

# 

# 

# 

# 

# 

# 