



International
Institute of Information
Technology Bangalore

SOFTWARE TESTING CS 731 PROJECT REPORT

By:

I.Cherish Chowdary (IMT2017022)

S.Sanjay Kumar (IMT2017037)

1 Introduction

The project we used for testing is a program to calculate different values of triangles like area, perimeter, radius of incircle etc. and check properties like congruency.

CodeBase: **Triangles.java**

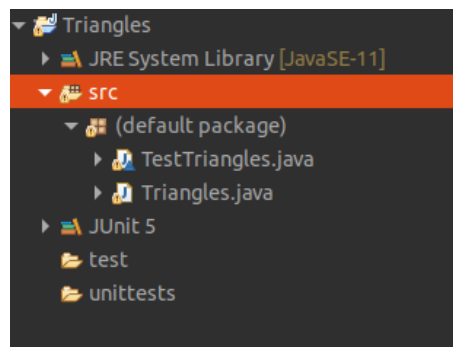
Testing tools used: JUnit and Pitclipse in Eclipse IDE

We used JUnit as the testing framework and Pitclipse to create mutants based on several different mutation operators and run test cases over these mutants.

All the test cases designed are in the file **TestTriangles.java**. The code contains test cases and expected output for each function.

2 Procedure Followed

- Import the project into eclipse and the files hierarchy should be as below figure.




- Use Junit to generate basic testing framework.
- Write test cases in the framework.
- Install Pitclipse plugin in eclipse for pitest mutation testing.
- Now run the test file using pitclipse to check mutation coverage.


3 Observations

3.1 Killed Mutations


1. Removed Sort - The inputs to some functions should be given as arrays(lengths of sides or angles). These functions used the sort function to sort these arrays. Initially, the test cases we provided were already in sorted order. So calling the sort function was useless. So we provided the test cases so as to kill this mutant.

 18: removed call to Triangles::sort


2. Changed Conditional boundary - This mutation shifts the boundary of relational operators by one i.e. changes ' $>$ ' to ' $>=$ ' etc. So adding test cases with boundary values killed this mutant.

 269: changed conditional boundary


3. Negated Conditional - This mutation inverts the conditional operators i.e. ' $>$ ' to ' $<=$ ' etc. Including all sets of test cases killed this mutant.


 22: negated conditional

4. Replaced operators - This mutation interchanges operators i.e. division with multiplication and addition with subtraction. Removing the test cases such that the values are not 0s or 1s, killed these mutants.

 243: Replaced double division with multiplication

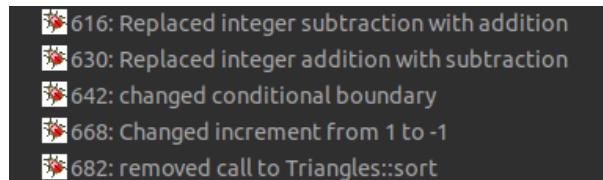
5. Changed increment and Replaced return values- This mutation changes the increment values from 1 to -1 or -1 to 1. We killed this mutant by exhausting the test cases.

 472: Changed increment from 1 to -1

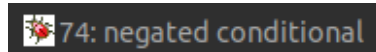
 228: replaced int return with 0 for Triangles::checkValidity

3.2 Survived Mutations

1. Sort function - We implemented merge sort to sort the arrays. Since the length of arrays is always 3, the values of some variables were 0. So we could not kill the 'Replaced operator' mutant here. Similarly, the recursive 'sort' statement was sometimes removed since there was only one element within it.



2. Negated conditions and boundary changes - Some of our functions used the OR operator to check certain conditions. Since only one true is required, the other statement could be negated.



4 Results

We got a line coverage of almost 100% and mutation coverage of 78%.

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage	Mutation Coverage
2	100% <div><div>566/568</div></div>	78% <div><div>356/454</div></div>

Breakdown by Package

Name	Number of Classes	Line Coverage	Mutation Coverage
default	2	100% <div><div>566/568</div></div>	78% <div><div>356/454</div></div>

Report generated by [PIT](#) 1.4.11

The below figure shows the number of mutants killed and survived. Most of the mutants survived are either negated conditions or boundary operators, because of the OR operator.

