

# Prática de Teste de Mutação

Alunos Rafael Rampim Soratto e Victor Daniel Pires

28 de outubro de 2022

**Problema 1.** Considere o método *findval* e um mutante gerado e responda as questões a seguir:

**Função original:**

```
1  /*
2  Efeitos:
3  Uma excecao NullPointerException e lancada se o parametro numbers for null.
4  A funcao retorna -1 if val nao estiver em numbers[].
5  Em outros casos a funcao retorna a ultima ocorrencia de val em numbers[]
6  */
7  public static int findVal(int numbers[], int val){
8      int find = -1;
9      for (int i=0; i<numbers.length(); i++){
10         if (numbers[i]==val){
11             find = i;
12         }
13     }
14     return(find)
15 }
```

**Mutante:**

```
1  public static int findVal(int numbers[], int val){
2      int find = -1;
3      for (int i=(0+1); i<numbers.length(); i++){ // Mutant
4          if (numbers[i]==val){
5              find = i;
6          }
7      }
8      return(find)
9  }
```

a) Crie um caso de teste que "mata" o mutante.

**Resposta:** O caso de teste que mata o mutante é aquele onde é passado um vetor vazio como parâmetro. Desta maneira o *i* começa com o valor 1, e na linha 4 ao acessar o array vazio na segunda posição (*numbers[i]*) provavelmente o programa irá parar de funcionar e levantar um null pointer exception. Portanto, o caso de teste *expect(findVal([],1).toBe(-1))* falha neste mutante pois acessa uma posição inválida do array.

b) Crie um caso de teste que não "mata" o mutante.

**Resposta:** Qualquer caso de teste onde um array não vazio é recebido por parâmetro não eliminará o mutante. Por exemplo: `expect(findVal([0,2],1).toBeEqual(-1)` ou então `expect(findVal([1,0],1).toBeEqual(0)`

c) Considerando a versão original do programa, crie um mutante equivalente e explique porque ele é equivalente.

**Problema 2.** Considere o método triangulos a seguir e faça o que se pede.

```
1 public static void triangulos(int x, int y, int z) {
2     if !(x < y + z || y < x + z || z < x + y) {
3         return(0); // "Nao e um triangulo"
4     }
5     if (x == y && y == z) {
6         return(1); // Triangulo Equilatero
7     } else if ((x == y && x!=z) || (x == z && x!= z)) {
8         return(2); // Triangulo Isosceles
9     } else if (x != y && x != z & y != z) {
10        return(3); //Triangulo Escaleno
11    }
12 }
```

a) Use a ferramenta de teste de mutação PIT (<https://pitest.org/>) para gerar mutantes e calcular o escore de mutação dos casos de teste implementados na atividade prática de Teste Funcional.

**Resposta:** Foi seguido o seguinte passo a passo para a configuração do pitest <sup>1</sup>:

1. Criação do projeto:

```
1 mvn archetype:generate -DgroupId=rafael -DartifactId=triangulo -
    DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion
    =1.4 -DinteractiveMode=false
```

2. cd triangulo

3. mvn package

4. mvn org.pitest:pitest-maven:mutationCoverage

Uma observação é que realizei algumas mudanças na classe triângulo da atividade passada adicionando a funcionalidade de retornar se é um triângulo ou não além do seu tipo. Os If's também estavam com erros e foram corrigidos para o programa funcionar corretamente.

Os resultados foram os seguintes:

---

<sup>1</sup>[https://www.youtube.com/watch?v=u3u\\_I09Y9S4](https://www.youtube.com/watch?v=u3u_I09Y9S4)

```

> NO_COVERAGE 3
=====
- Timings
=====
> pre-scan for mutations : < 1 second
> scan classpath : < 1 second
> coverage and dependency analysis : < 1 second
> build mutation tests : < 1 second
> run mutation analysis : < 1 second
=====
> Total : < 1 second
=====
- Statistics
=====
>> Line Coverage: 12/19 (63%)
>> Generated 30 mutations Killed 14 (47%)
>> Mutations with no coverage 9. Test strength 67%
>> Ran 21 tests (0.7 tests per mutation)
Enhanced functionality available at https://www.arcmutate.com/
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.815 s
[INFO] Finished at: 2022-10-27T02:17:06-03:00
[INFO] -----
rafael@rafael-IdeaPad-Gaming-3-15IMH05:~/Documentos/workspace/teste-mutantes/triangulos$

```

Figura 1: Resultado parte 1: execução do comando

## Pit Test Coverage Report

### Project Summary

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
1	63% <div><div>12/19</div></div>	47% <div><div>14/30</div></div>	67% <div><div>14/21</div></div>

### Breakdown by Package

Name	Number of Classes	Line Coverage	Mutation Coverage	Test Strength
<a href="#">rafael</a>	1	63% <div><div>12/19</div></div>	47% <div><div>14/30</div></div>	67% <div><div>14/21</div></div>

Report generated by [PIT](#) 1.9.9

Enhanced functionality available at [arcmutate.com](https://www.arcmutate.com)

Figura 2: Resultado parte 2

# Pit Test Coverage Report

## Package Summary

rafael

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
1	63% <div><div></div><div>12/19</div></div>	47% <div><div></div><div>14/30</div></div>	67% <div><div></div><div>14/21</div></div>

## Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
<a href="#">App.java</a>	63% <div><div></div><div>12/19</div></div>	47% <div><div></div><div>14/30</div></div>	67% <div><div></div><div>14/21</div></div>

Report generated by [PIT](#) 1.9.9

Figura 3: Resultado parte 3

## App.java

```
1 package rafael;
2
3 /**
4  * Hello world!
5  *
6  */
7 public class App
8 {
9     public static String triangulos(int x, int y, int z) {
10
11         // verificando se é um triangulo
12 9         if (x < y + z || y < x + z || z < x + y) {
13 1         System.out.println("Trata-se de um triangulo");
14         } else {
15 1         return "Não é um triângulo";
16         }
17
18         // analisando o tipo de triangulo
19 2         if (x == y && y == z) {
20 1         System.out.println("Lados: " + x + "," + y + "," + z);
21 1         return "Três lados iguais. Triângulo Equilátero\n";
22
23 4         } else if ((x == y && x != z) || (x == z && x != y)) {
24 1         System.out.println("Lados: " + x + "," + y + "," + z);
25 1         return "Dois lados iguais. Triângulo Isosceles\n";
26
27 5         } else if (x != y && x != z && y != z) {
28 1         System.out.println("Lados: " + x + "," + y + "," + z);
29 1         return "Três lados diferentes. Triângulo Escaleno\n";
30         } else {
31 1         return "Não é um triângulo";
32         }
33     }
34
35     public static void main(String[] args) {
36 1         System.out.println("Teste do Triângulo \n\n");
37         String r1 = App.triangulos(2, 2, 2);
38         String r2 = App.triangulos(2, 2, 3);
39         String r3 = App.triangulos(2, 3, 4);
40     }
41 }
```

Figura 4: Resultado parte 4: refatoração da classe com melhor resultado para as mutações

41 }

## Mutations

	1. changed conditional boundary → SURVIVED
	2. changed conditional boundary → NO_COVERAGE
	3. changed conditional boundary → NO_COVERAGE
	4. Replaced integer addition with subtraction → SURVIVED
12	5. Replaced integer addition with subtraction → NO_COVERAGE
	6. Replaced integer addition with subtraction → NO_COVERAGE
	7. negated conditional → SURVIVED
	8. negated conditional → NO_COVERAGE
	9. negated conditional → NO_COVERAGE
13	1. removed call to java/io/PrintStream::println → SURVIVED
15	1. replaced return value with "" for rafael/App::triangulos → NO_COVERAGE
19	1. negated conditional → KILLED
	2. negated conditional → KILLED
20	1. removed call to java/io/PrintStream::println → SURVIVED
21	1. replaced return value with "" for rafael/App::triangulos → KILLED
	1. negated conditional → KILLED
23	2. negated conditional → KILLED
	3. negated conditional → KILLED
	4. negated conditional → NO_COVERAGE
24	1. removed call to java/io/PrintStream::println → SURVIVED
25	1. replaced return value with "" for rafael/App::triangulos → KILLED
	1. Replaced bitwise AND with OR → KILLED
	2. negated conditional → KILLED
27	3. negated conditional → KILLED
	4. negated conditional → KILLED
	5. negated conditional → KILLED
28	1. removed call to java/io/PrintStream::println → SURVIVED
29	1. replaced return value with "" for rafael/App::triangulos → KILLED
31	1. replaced return value with "" for rafael/App::triangulos → KILLED
36	1. removed call to java/io/PrintStream::println → NO_COVERAGE

## Active mutators

- CONDITIONALS\_BOUNDARY
- EMPTY\_RETURNS
- FALSE\_RETURNS
- INCREMENTS
- INVERT\_NEGS
- MATH
- NEGATE\_CONDITIONALS
- NULL\_RETURNS
- PRIMITIVE\_RETURNS
- TRUE\_RETURNS
- VOID\_METHOD\_CALLS

## Tests examined

- rafael.AppTest.shouldAnswerWithTrue(rafael.AppTest) (4 ms)

Report generated by PIT 1.9.9

**Figura 5:** Resultado parte 5: mutações geradas

a) Escreva casos de teste para "matar" pelo menos mais um mutante, se possível. **Resposta:**

```
1      assertEquals(App.triangulos(3,4,5), "Tres lados diferentes. Triangulo
      Escaleno\n");
2      assertEquals(App.triangulos(1,1,1), "Tres lados iguais. Triangulo
      Equilatero\n");
3      assertEquals(App.triangulos(-1,4,4), "Nao e um triangulo");
4      assertEquals(App.triangulos(3,3,4), "Dois lados iguais. Triangulo
      Isosceles\n");
```

b) Identifique se possível algum mutante equivalente.

**Resposta:** Apresentado na Figura 4.

- c) Reimplemente a função triângulos em Javascript e use a ferramenta Stryker para gerar mutantes. Após isso, implemente casos de teste para obter o mesmo escore de mutação do item 2.(a).

**Resposta:** A implementação da classe triângulo em javascript na Figura 7:

```
1  function triangle([x, y, z]) {
2      // verificando se é um triângulo
3      if (x < y + z || y < x + z || z < x + y) {
4          // analisando o tipo de triângulo
5          if (x == y && y == z) {
6              return "Três lados iguais. Triângulo Equilátero\n";
7          }
8          else if ((x == y) || (x == z) || (y == z)) {
9              return "Dois lados iguais. Triângulo Isosceles\n";
10             }
11             else if (x != y && x != z && y != z) {
12                 return "Três lados diferentes. Triângulo Escaleno\n";
13             }
14             else {
15                 return "Não é um triângulo";
16             }
17         }
18     }
19 }
20
21 module.exports = {
22     triangle
23 };
```

**Figura 6:** Implementação da classe triângulo em javascript.

A implementação da classe de teste do triângulo em javascript na Figura 7:

```

1  const { triangle } = require('./triangle.js')
2
3  describe('Unit test for Triangle', () => {
4    test('Escaleno', () => {
5      let array = [1, 2, 3];
6      expect(triangle(array)).toEqual("Três lados diferentes. Triângulo Escaleno\n")
7    })
8
9    test('Equilatero', () => {
10     let array = [1, 1, 1];
11     expect(triangle(array)).toEqual("Três lados iguais. Triângulo Equilatero\n")
12   })
13
14   test('Isosceles', () => {
15     let array = [1, 2, 2];
16     expect(triangle(array)).toEqual("Dois lados iguais. Triângulo Isosceles\n")
17   })
18
19   test('Not triangle', () => {
20     let array = [0, 0, 0];
21     expect(triangle(array)).toEqual("Não é um triângulo")
22   })
23 })

```

**Figura 7:** Implementação da classe triângulo em javascript.

Comandos necessários para criar o stryker no projeto:

1. "npm install"
2. "npm i -g stryker-cli"
3. "./node\_modules/@stryker-mutator/core/bin/stryker.js init"
  - Utilizar a conf:

```

1          ``? Do you want to install Stryker locally?: npm
2  ? Are you using one of these frameworks? Then select a preset
   configuration. None/other
3  ? Which test runner do you want to use? If your test runner isn't
   listed here, you can choose "command" (it uses your npm testcommand
   , but will come with a big performance penalty) jest
4  ? What kind of code do you want to mutate? javascript
5  ? [optional] What kind transformations should be applied to your code?
   (Press to select, to toggle all, to invert selection)
6  ? Which reporter(s) do you want to use? html, clear-text, progress
7  Note: Use spacebar for multiple selection or choose html and press
   enter
8  ? Which package manager do you want to use? npm
9  ``

```

4. "./node\_modules/@stryker-mutator/core/bin/stryker.js run"

Os resultados das mutações da classe triângulo no javascript estão nas Figuras [8](#) [9](#) [10](#):

Ran 1.92 tests per mutant on average.						
File	% score	# killed	# timeout	# survived	# no cov	# error
All files	64.41	38	0	19	2	0
triangle.js	64.41	38	0	19	2	0

Figura 8: Resultado da mutação triangulo.js

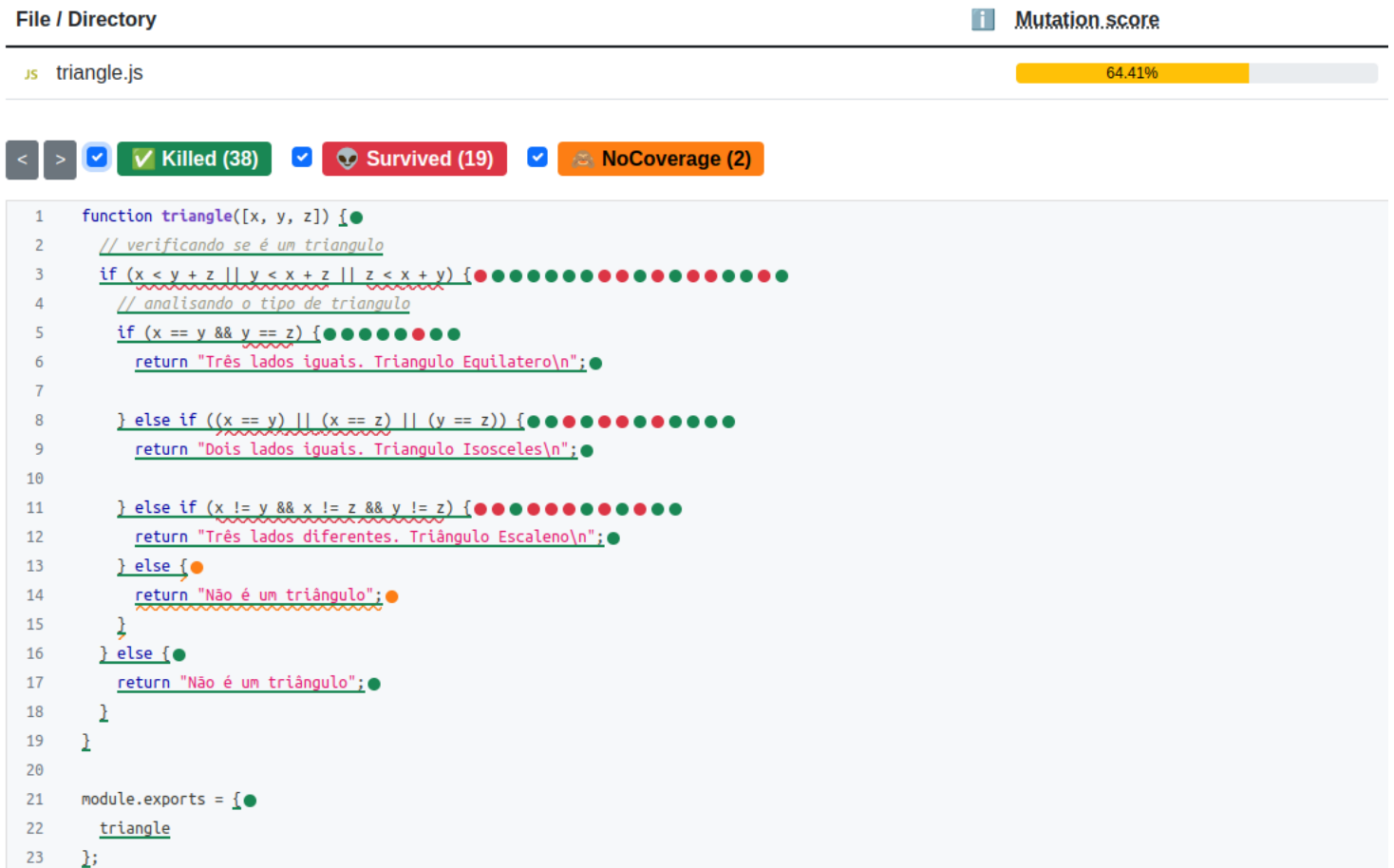


Figura 9: Resultado da mutação triangulo.js em formato de html



File / Directory	 Mutation score	# Killed	# Survived	# Timeout	# No coverage	# Ignored	# Runtime errors	# Compile errors	Total detected	Total undetected	Total mutants	
 All files	<div><div></div></div> 64.41%	64.41	38	19	0	2	0	0	0	38	21	59
js <a href="#">triangle.js</a>	<div><div></div></div> 64.41%	64.41	38	19	0	2	0	0	0	38	21	59

Figura 10: Resultado da mutação triangulo.js em formato de html

- e) Reimplemente a função triângulos em Python e use a ferramenta mutmut ou mutatest para gerar mutantes. Após isso, implemente casos de teste para obter o mesmo escore de mutação do item 2.(a).