Atividade 04 - Testes de Mutação

Mateus Miranda de Frias - 8516837

IDE: Intellij;
Reporte de cobertura de mutação: Pit Test

Após alguns problemas de configuração, consegui rodar o Pitest indicando a versão mais recente (1.6.7):

```
<plugins>
    <plugin>
        <groupId>org.pitest</groupId>
        <artifactId>pitest-maven</artifactId>
        <version>1.6.7</version>
    </plugin>
</plugins>
```

Nas atividades anteriores, usei bastante a técnica de valor limite para criação de casos de teste. Com isso, após instalar o Pitest e rodar para as cinco classes, descobri que os cenários de teste já apresentavam bons valores de cobertura de mutação, com 3 das 5 classes apresentando 100% de cobertura:

Pit Test Coverage Report

Package Summary

codigos

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
5	100%	66/66	90%	64/71	90%	64/71

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
ContaCorrente.java	100%	16/16	100%	9/9	100%	9/9
Identifier.java	100%	12/12	100%	12/12	100%	12/12
Pedido.java	100%	9/9	92%	11/12	92%	11/12
Triangulo.java	100%	12/12	71%	15/21	71%	15/21
Words.java	100%	17/17	100%	17/17	100%	17/17

Report generated by PIT 1.6.7

```
- Statistics
============================================================
>> Line Coverage: 68/68 (100%)
>> Generated 71 mutations Killed 64 (90%)
>> Mutations with no coverage 0. Test strength 90%
>> Ran 171 tests (2.41 tests per mutation)
```

Os mutantes vivos das duas classes restantes:

Pedido.java

```
1   package codigos;
2
3   public class Pedido {
4
5
6       public float calculaTaxaDesconto(boolean primeiraCompra, String tipoCliente, float valorCompra) {
7           float taxa = 0;
8   3       if (valorCompra >= 500 || tipoCliente.equals("ouro"))
9   1           return 15;
10          else
11  4       if(tipoCliente.equals("prata") || primeiraCompra || valorCompra >= 400)
12              taxa = 10;
13          else
14  3       if(valorCompra >= 200 || tipoCliente == "bronze")
15              taxa = 5;
16  1       return taxa;
17      }
18  }
```

Mutations

```
    1. changed conditional boundary → SURVIVED
8   2. negated conditional → KILLED
    3. negated conditional → KILLED
9   1. replaced float return with 0.0f for codigos/Pedido::calculaTaxaDesconto → KILLED
    1. changed conditional boundary → KILLED
```

Para esta classe, não testei o valor limite 500 na condição em questão. O caso de teste abaixo fez a cobertura atingir 100%:

```
@Test
public void testTaxa15PeloValorCompraNoLimite(){
    float taxa = pedido.calculaTaxaDesconto(false,"",500);
    assertEquals(15,taxa,0);
}
```

Triangulo.java

```
1   package codigos;
2
3   public class Triangulo {
4
5       public static String classificaTriangulo(int LA, int LB, int LC)
6           String resposta="";
7  6        if (LA<=0 || LB <=0 || LC <0)
8               throw new LadoInvalidoException("lado invalido");
9
10 9        if ( (LA >= LB + LC) || (LB >= LA + LC) || (LC > LA + LB))
11              resposta = "NAO FORMA TRIANGULO";
12          else
13          {
14 2            if (LA==LB && LB==LC)
15                  resposta = "EQUILATERO";
16              else
17              {
18 3                if (LA==LB || LB==LC || LA==LC)
19                      resposta = "ISOSCELES";
20                  else
21                      resposta = "ESCALENO";
22              }
23          }
24 1        return resposta;
25      }
26  }
```

Mutations

```
1. changed conditional boundary → SURVIVED
2. changed conditional boundary → SURVIVED
3. changed conditional boundary → SURVIVED
4. negated conditional → KILLED
```

7

Para a condição de lado inválido, adicionei casos de teste para os valores limite 0:

```
@Test
public void testLadoAInvalidoLimite() throws
LadoInvalidoException{
    thrown.expect(LadoInvalidoException.class);
    thrown.expectMessage("lado invalido");
    Triangulo.classificaTriangulo(0,5,5);
}

@Test
```

```
public void testLadoBInvalidoLimite() throws
LadoInvalidoException{
    thrown.expect(LadoInvalidoException.class);
    thrown.expectMessage("lado invalido");
    Triangulo.classificaTriangulo(5,0,5);
}
```

Para a condição de “não forma triângulo”, também adicionei casos de teste para os valores limite onde LA = LB + LC e LB = LA + LC, conforme:

```
@Test
public void testNaoFormaTrianguloALimite() throws
LadoInvalidoException{
    String resultado = Triangulo.classificaTriangulo(10,5,5);
    assertEquals("NAO FORMA TRIANGULO",resultado);
}

@Test
public void testNaoFormaTrianguloBLimite() throws
LadoInvalidoException{
    String resultado = Triangulo.classificaTriangulo(5,10,5);
    assertEquals("NAO FORMA TRIANGULO",resultado);
}
```

Percebi que essas mutações não resolveram a cobertura nas duas linhas em questão, então imaginei que a operação de mutação “changed conditional boundary” não estava mudando somente >= para = nas condições dos lados A e B, mas mudou também > para = nos testes do Lado C.

Esses cenários não caem em uma exceção, pois o código aceita LC = 0 e LC = LA + LB. Creio que isso seja um defeito pois os valores não formam triângulo, mas mantive essas condições no código conforme versão original, já que gerou essas formas interessantes de matar alguns mutantes.

Para cobrir essa mutação, criei um caso de teste de triângulo isósceles que contemplasse a condição LC = LA + LB, e um caso de teste de lado inválido entrando com LC = 0 e LA > LB + LC:

```
@Test
public void testIsoscelesBCLimite() throws  LadoInvalidoException{
    String resultado = Triangulo.classificaTriangulo(5,5,10);
    assertEquals("ISOSCELES",resultado);
}

@Test
public void testNaoFormaTrianguloCZero() throws
LadoInvalidoException{
```

```
    String resultado = Triangulo.classificaTriangulo(11,5,0);
    assertEquals("NAO FORMA TRIANGULO",resultado);
}
```

Com os novos casos de testes adicionados às classes, a cobertura de mutação chegou em 100%:

Pit Test Coverage Report

Package Summary

codigos

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
5	100%	66/66	100%	71/71	100%	71/71

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
ContaCorrente.java	100%	16/16	100%	9/9	100%	9/9
Identifier.java	100%	12/12	100%	12/12	100%	12/12
Pedido.java	100%	9/9	100%	12/12	100%	12/12
Triangulo.java	100%	12/12	100%	21/21	100%	21/21
Words.java	100%	17/17	100%	17/17	100%	17/17

Report generated by PIT 1.6.7

```
- Statistics
============================================================
>> Line Coverage: 68/68 (100%)
>> Generated 71 mutations Killed 71 (100%)
>> Mutations with no coverage 0. Test strength 100%
>> Ran 158 tests (2.23 tests per mutation)
```