U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Teste, Verificação e Validação de Software

Correção do Exercício Prático

João Bordalo – ei12040
João Pereira – ei12023

4 de outubro de 2016

1 GDC

a) Através da ferramenta *Pitclipse* identifique os mutantes presentes na função ***GDC*** da classe ***Algorithm*****.**

```
public int gcd(int x, int y) {
    int tmp;
    while (y != 0) {
        tmp = x % y;
        x = y;
        y = tmp;
    }
    return x;
}
```

Figura 1 - Função gdc

b) Encontre a melhor solução com o método de eliminação de mutantes.

c) Teste a sua solução com a ferramenta *Pitclipse* e verifique se é possível eliminar os mutantes.

2 Min

a) Através da ferramenta *Pitclipse* identifique os mutantes presentes na função ***Min*** da classe ***Algorithm*****.**

```
public int Min(int x, int y) {
    int v;

    if (x < y)
        v = x;
    else
        v = y;

    return v;
}
```

Figura 2 - Função min

b) Encontre a melhor solução com o método de eliminação de mutantes e verifique com a ferramenta *Pitclipse* se é possível eliminar os mutantes.

c) O que conclui dos mutantes da função ***Min*** da classe ***Algorithm*****?**

3 NumZero e NegateArray

a) Através da ferramenta *Pitclipse* identifique os mutantes presentes na função ***NumZero*** e ***NegateArray*** da classe ***Algorithm*** e encontre possíveis soluções de teste capazes de os eliminar.

```
public int numZero(int[] x) {
    int count = 0;

    for (int i = 0; i < x.length; i++)
        if (x[i] == 0)
            count++;

    return count;
}

public void negateArray(final float i, float a[]) {
    for(int k = 0; k < a.length; k++)
        a[k] = a[k] * (-i);
}
```

Figura 3 - Função numZero e negateArray

4 IsLessThanThree

a) A função ***IsLessThanThree*** já tem um teste associado, porém mesmo com um ***assert*** aplicado e estando correto, os mutantes não morreram.

Por favor corrija a situação.

```
public boolean isLessThanThree(int number) {
    return (number < 3);
}
```

Figura 4 - Função isLessThanThree

```
@Test
public void testLessThanThree() {
    assertTrue(this.alg.isLessThanThree(2));
}
```

Figura 5 - Teste da função

Mutantes Vivos

changed conditional boundary

replaced return of integer sized value with (x == 0 ? 1 : 0)

Bom trabalho!

1 GDC - correção

a)

- Negate Conditionals Mutator, na linha 6 (!= → ==)
- Math Mutator, na linha 7 (% → *)
- Return Values Mutator, na linha 11 (return value is 0)

```
- Mutators
================================================================================
> org.pitest.mutationtest.engine.gregor.mutators.ReturnValsMutator
>> Generated 1 Killed 1 (100%)
> KILLED 1 SURVIVED 0 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 0
--------------------------------------------------------------------------------
> org.pitest.mutationtest.engine.gregor.mutators.MathMutator
>> Generated 1 Killed 1 (100%)
> KILLED 1 SURVIVED 0 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 0
--------------------------------------------------------------------------------
> org.pitest.mutationtest.engine.gregor.mutators.NegateConditionalsMutator
>> Generated 1 Killed 1 (100%)
> KILLED 1 SURVIVED 0 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 0
--------------------------------------------------------------------------------
```

b)

O mutante Return Values Mutator pode ser eliminado obrigando ao *coverage* daquela linha. Resta apenas analisar os outros mutantes.

Possível solução:

	x	**y**	**gcd**	**!= --> ==**	**% → ***
T1	0	0	0	loop	0
T2	0	1	1	0	1
T3	1	0	1	-	1
T3	2	1	1	2	2

A solução x=2 e y=1 é um bom caso de teste.

c)

```java
/**
 * Test gcd function
 */
@Test
public void TestGCD() {
    assertEquals(1, this.alg.gcd(2, 1));
}
```

2 Min - correção

a)

- Conditional Boundary Mutator, linha 17 (< → <=)
- Negate Conditionals Mutator, linha 17 (< → >=)
- Return Values Mutator, na linha 22 (return value is 0)

```
- Mutators
================================================================================
> org.pitest.mutationtest.engine.gregor.mutators.ConditionalsBoundaryMutator
>> Generated 1 Killed 0 (0%)
> KILLED 0 SURVIVED 0 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 1
--------------------------------------------------------------------------------
> org.pitest.mutationtest.engine.gregor.mutators.ReturnValsMutator
>> Generated 1 Killed 0 (0%)
> KILLED 0 SURVIVED 0 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 1
--------------------------------------------------------------------------------
> org.pitest.mutationtest.engine.gregor.mutators.NegateConditionalsMutator
>> Generated 1 Killed 0 (0%)
> KILLED 0 SURVIVED 0 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 1
```

b)

	x	y	min	< → <=	< → >=
T1	0	0	0	0	0
T2	0	1	0	0	1
T3	1	0	0	0	1

Neste caso não existe nenhum teste que consiga matar o mutante <= visto que o resultado deste teste mutado é igual ao resultado do teste original.

c)

O mutante que não se consegue matar na função é um **mutante funcionalmente equivalente**.

3 NumZero e NegateArray - correção

Neste exercício não é pedido que se faça a resolução pelo método de eliminação de mutantes pelo que bastaria apresentar testes unitários que matassem os mutantes.

```
/**
 * Test numZero function
 */
@Test
public void TestNumZero() {
    int a[] = {0};

    assertEquals(1, this.alg.numZero(a));
}

/**
 * Test negateArray function
 */
@Test
public void TestNegateArray() {
    float[] a = {1, 2, 4};
    float b[] = new float[a.length];

    System.arraycopy(a, 0, b, 0, a.length);

    this.alg.negateArray(2, a);

    for(int i = 0; i < a.length; i++)
        assertEquals(b[i] * (-2), a[i], floatTolerance);
}
```

4 IsLessThanThree - correção

Neste exercício bastaria acrescentar um teste que consiga passar no teste mutante mas não no teste original. Por exemplo:

```
@Test
public void testLessThanThree2() {
    assertFalse(this.alg.isLessThanThree(3));
}
```