

Aluno: Marco Antonio Cottorello Henry

1) ContaCorrenteTest

```
package codigos;

import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;
import org.junit.rules.ExpectedException;
import static org.junit.Assert.assertEquals;

public class ContaCorrenteTest {

    @Rule
    public ExpectedException thrown = ExpectedException.none();

    private ContaCorrente conta;

    @Before
    public void setUp() {
        conta = new ContaCorrente(500, 100);
    }

    @Test
    public void testSaqueValorZero() throws Exception {
        thrown.expect(Exception.class);
        thrown.expectMessage("Valor invalido");
        ContaCorrente cc = new ContaCorrente(100, 100);
        cc.saque(0);
    }

    @Test
    public void testSaqueValorNegativo() throws Exception {
        thrown.expect(Exception.class);
        thrown.expectMessage("Valor invalido");
        conta.saque(-50);
    }
}
```

```

@Test
public void testSaqueComSucessoSaldoPositivoSemLimiteUsado()
throws Exception {

    ContaCorrente cc = new ContaCorrente(500, 0);
    float novoSaldo = cc.saque(100);
    assertEquals(400, novoSaldo, 0.001);
    assertEquals(400, cc.getSaldo(), 0.001);
}

@Test
public void testSaqueComSucessoUsandoLimiteSaldoNegativo()
throws Exception {

    ContaCorrente cc = new ContaCorrente(200, 200);
    float novoSaldo = cc.saque(300);
    assertEquals(-100, novoSaldo, 0.001);
    assertEquals(-100, cc.getSaldo(), 0.001);
}

@Test
public void testSaqueExatoSaldoMaisLimite() throws Exception {
    ContaCorrente cc = new ContaCorrente(100, 50); //
    Saldo+Limite = 150
    float novoSaldo = cc.saque(150);
    assertEquals(-50, novoSaldo, 0.001);
    assertEquals(-50, cc.getSaldo(), 0.001);
}

@Test
public void testSaqueExatoSaldoSemLimite() throws Exception {
    ContaCorrente cc = new ContaCorrente(100, 0); //
    Saldo+Limite = 100
    float novoSaldo = cc.saque(100);
    assertEquals(0, novoSaldo, 0.001);
    assertEquals(0, cc.getSaldo(), 0.001);
}

@Test
public void testSaqueSaldoInsuficiente() throws Exception {

```

```

        thrown.expect(Exception.class);
        thrown.expectMessage("Saldo Insuficiente");
        ContaCorrente cc = new ContaCorrente(200, 100); //
Saldo+Limite = 300
        cc.saque(400);
    }

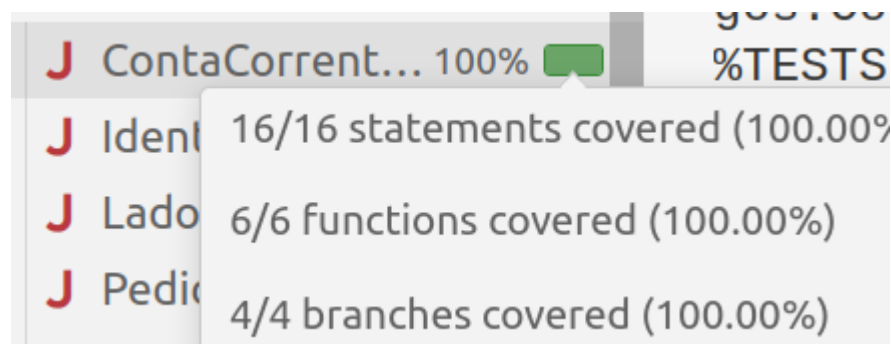
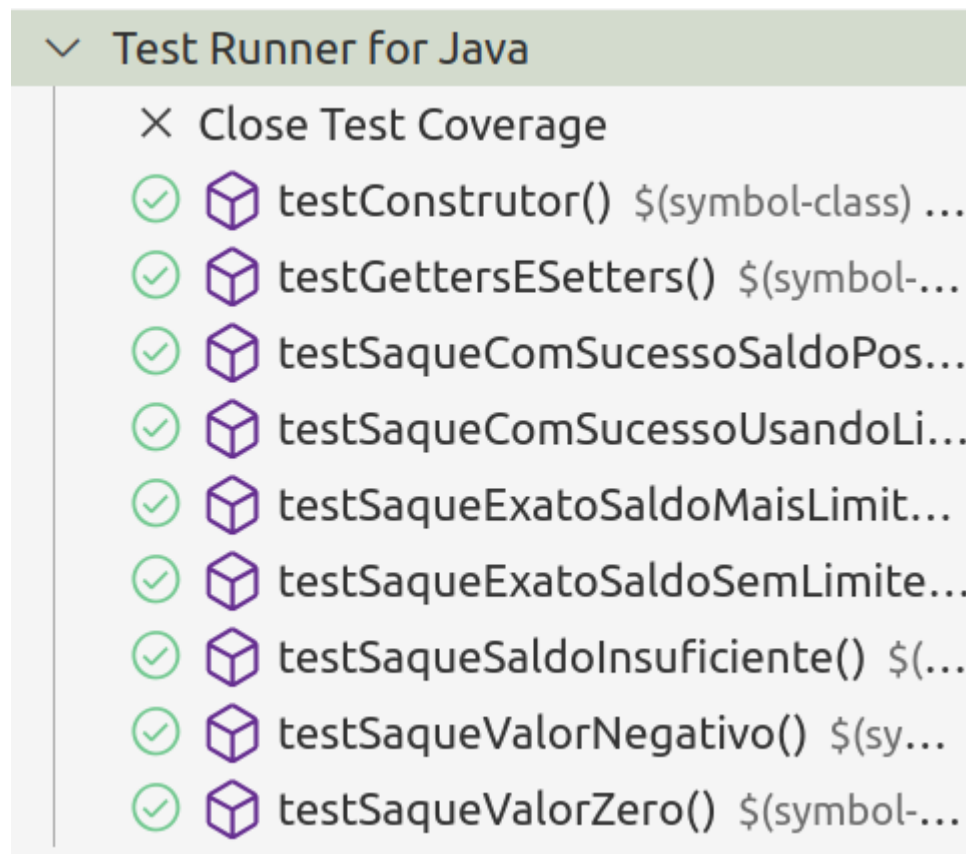
    @Test
    public void testGettersESetters() {
        ContaCorrente cc = new ContaCorrente(0, 0);

        cc.setSaldo(1000);
        assertEquals(1000, cc.getSaldo(), 0.001);

        cc.setLimite(500);
        assertEquals(500, cc.getLimite(), 0.001);
    }

    @Test
    public void testConstrutor() {
        ContaCorrente cc = new ContaCorrente(750, 250);
        assertEquals(750, cc.getSaldo(), 0.001);
        assertEquals(250, cc.getLimite(), 0.001);
    }
}

```



2) IdentifierTest

```
package codigos;
```

```
import org.junit.Assert;  
import org.junit.Rule;  
import org.junit.Test;  
import org.junit.rules.ExpectedException;
```

```
public class IdentifierTest {
```

```
    @Rule
```

```
public ExpectedException thrown = ExpectedException.none();

@Test
public void testIDValido() {
    Assert.assertTrue(Identifier.validaIdentificador("a1"));
}

@Test
public void testIDValidoSimples() {
    Assert.assertTrue(Identifier.validaIdentificador("abc"));
}

@Test
public void testIDValidomenortam() {
    Assert.assertTrue(Identifier.validaIdentificador("a"));
}

@Test
public void testIDValidomaiortam() {
    Assert.assertTrue(Identifier.validaIdentificador("abcdef"));
}

@Test
public void testIDValidomaiortamComNumero() {
    Assert.assertTrue(Identifier.validaIdentificador("abcde1"));
}

@Test
public void testIDInvalidoNull() {
    Assert.assertFalse(Identifier.validaIdentificador(null));
}

@Test
public void testIDInvalidovazio() {
    Assert.assertFalse(Identifier.validaIdentificador(""));
}

@Test
public void testIDInvalidoLongo() {
```

```
Assert.assertFalse(Identifier.validaIdentificador("abcdefg"));
    }

    @Test
    public void testIDInvalidoiniciodigito() {

Assert.assertFalse(Identifier.validaIdentificador("1abc"));
    }















    @Test
    public void testIDInvalidocaracterediferente() {


Assert.assertFalse(Identifier.validaIdentificador("ab-c"));
    }

    @Test
    public void testIDInvalidoespaço() {
        Assert.assertFalse(Identifier.validaIdentificador("ab
c"));
    }
}
```

Test Runner for Java

× Close Test Coverage

- ✓  testIDValido() \$(symbol-class)...
- ✓  testIDInvalidoNull() \$(symbol-...
- ✓  testIDValidoSimples() \$(sym...
- ✓  testIDValidomenortam() \$(s...
- ✓  testIDValidomaiortam() \$(sy...
- ✓  testIDValidomaiortamComN...
- ✓  testIDInvalidovazio() \$(symb...
- ✓  testIDInvalidoLongo() \$(sym...
- ✓  testIDInvalidoin...   
- ✓  testIDInvalidocaracteredifer...
- ✓  testIDInvalidoespaço() \$(sym...

J Identifier.java 100%  %TESTE

J Lado 12/12 statements covered (100.00%)

J Pedido 2/2 functions covered (100.00%)

J Triangulo 14/14 branches covered (100.00%)

3) PedidoTest

```
package codigos;

import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
```

```
public class PedidoTest {

    private Pedido pedido;

    @BeforeClass
    public static void setupClass() {

    }

    @Before
    public void setup() {

        pedido = new Pedido();
    }

    @Test
    public void testTaxaZero() {
        float taxa = pedido.calculaTaxaDesconto(false, "", 100);
        assertEquals(0, taxa, 0.001);
    }

    @Test
    public void testTaxa15PeloValorCompraMaior500() {
        float taxa = pedido.calculaTaxaDesconto(false, "", 600);
        assertEquals(15, taxa, 0.001);
    }

    @Test
    public void testTaxa15PeloValorCompraIgual500() {
        float taxa = pedido.calculaTaxaDesconto(false, "", 500);
        assertEquals(15, taxa, 0.001);
    }

    @Test
    public void testTaxa15PeloTipoClienteOuro() {
        float taxa = pedido.calculaTaxaDesconto(false, "ouro", 300);
        assertEquals(15, taxa, 0.001);
    }

    @Test
    public void testTaxa10PeloTipoClientePrata() {
```



```

        // tipoCliente "prata", valor < 400, nao primeiraCompra
        float taxa = pedido.calculaTaxaDesconto(false, "prata", 300);
        assertEquals(10, taxa, 0.001);
    }

    @Test
    public void testTaxa10PelaPrimeiraCompra() {
        // primeiraCompra, tipoCliente "", valor < 400
        float taxa = pedido.calculaTaxaDesconto(true, "", 300);
        assertEquals(10, taxa, 0.001);
    }

    @Test
    public void testTaxa10PeloValorCompraMaiorIgual400() {
        // valorCompra >= 400, < 500, nao "ouro", nao "prata", nao
        primeiraCompra
        float taxa =
        pedido.calculaTaxaDesconto(false, "bronze", 400);
        assertEquals(10, taxa, 0.001);
    }

    @Test
    public void
    testTaxa10PeloValorCompraMaiorIgual400ClienteNormal() {
        // valorCompra >= 400, < 500, nao "ouro", nao "prata", nao
        primeiraCompra
        float taxa =
        pedido.calculaTaxaDesconto(false, "normal", 450);
        assertEquals(10, taxa, 0.001);
    }

    @Test
    public void testTaxa5PeloValorCompraMaiorIgual200() {
        // valorCompra >= 200, < 400, cliente nao "ouro", "prata",
        nao primeiraCompra
        float taxa = pedido.calculaTaxaDesconto(false, "", 250);
        assertEquals(5, taxa, 0.001);
    }

    @Test
    public void testTaxa5PeloTipoClienteBronze() {

```

```

        // tipoCliente "bronze" (note: uses == in implementation),
        valor < 200
        // nao primeiraCompra
        float taxa =
pedido.calculaTaxaDesconto(false, "bronze", 100);
        assertEquals(5, taxa, 0.001);
    }

    @Test
    public void
testTaxa5PeloTipoClienteBronzeComValorMaiorMasOutrasCondicoesDe10
NaoAtendidas() {
        // tipoCliente "bronze", valor >=200 but <400, not
        primeiraCompra
        float taxa =
pedido.calculaTaxaDesconto(false, "bronze", 250);
        assertEquals(5, taxa, 0.001); // Will hit the valorCompra >=
200 first
    }

    @Test
    public void testPrioridadeTaxa15OuroSobrePrata() {
        // tipoCliente "ouro" deve dar 15% mesmo se valor
        qualificaria para 10% por "prata"
        float taxa = pedido.calculaTaxaDesconto(false, "ouro",
450); // valor >= 400
        assertEquals(15, taxa, 0.001);
    }

    @Test
    public void testPrioridadeTaxa10PrataSobreBronze() {
        // tipoCliente "prata" deve dar 10% mesmo se valor
        qualificaria para 5% por "bronze"
        float taxa = pedido.calculaTaxaDesconto(false, "prata",
250); // valor >= 200
        assertEquals(10, taxa, 0.001);
    }















    @Test
    public void testPrioridadeTaxa10PrimeiraCompraSobreBronze() {

```

```
float taxa = pedido.calculaTaxaDesconto(true, "bronze",  
250); // primeiraCompra, valor >=200  
assertEquals(10, taxa, 0.001);  
}  
}
```

Test Runner for Java

× Close Test Coverage

- ✓  testPrioridadeTaxa10PrataS...
- ✓  testPrioridadeTaxa10Primeir...
- ✓  testPrioridadeTaxa15OuroSo...
- ✓  testTaxa10PelaPrimeiraCom...
- ✓  testTaxa10PeloTipoClienteP...
- ✓  testTaxa10PeloValorCompra...
- ✓  testTaxa10PeloValorCompra...
- ✓  testTaxa15PeloTipoClienteO...
- ✓  testTaxa15PeloValorCompra...
- ✓  testTaxa15PeloValorCompra...
- ✓  testTaxa5PeloTipoClienteBr...
- ✓  testTaxa5PeloTipoClienteBr...
- ✓  testTaxa5PeloValorCompra...
- ✓  testTaxaZero() \$(symbol-class...

J	Pedido.java	100%	100.00%
J	Triangulo.java	9/9 statements covered (100.00%)	100.00%
J	Words.java	2/2 functions covered (100.00%)	100.00%
>	resources	14/14 branches covered (100.00%)	100.00%

4)TrianguloTest

```
package codigos;

import org.junit.Rule;
import org.junit.Test;
import org.junit.rules.ExpectedException;
import static org.junit.Assert.assertEquals;

public class TrianguloTest {

    @Rule
    public ExpectedException thrown = ExpectedException.none();

    @Test
    public void testEquilatero() throws LadoInvalidoException {
        String resultado = Triangulo.classificaTriangulo(5, 5, 5);
        assertEquals("EQUILATERO", resultado);
    }

    @Test
    public void testIsoscelesAB() throws LadoInvalidoException {
        String resultado = Triangulo.classificaTriangulo(5, 5, 4);
        assertEquals("ISOSCELES", resultado);
    }

    @Test
    public void testIsoscelesAC() throws LadoInvalidoException {
        String resultado = Triangulo.classificaTriangulo(5, 4, 5);
        assertEquals("ISOSCELES", resultado);
    }
}
```

```
@Test
public void testIsoscelesBC() throws LadoInvalidoException {
    String resultado = Triangulo.classificaTriangulo(4, 5, 5);
    assertEquals("ISOSCELES", resultado);
}

@Test
public void testEscaleno() throws LadoInvalidoException {
    String resultado = Triangulo.classificaTriangulo(5, 6, 7);
    assertEquals("ESCALENO", resultado);
}

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

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

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

@Test
public void testLadoAInvalidoZero() throws
LadoInvalidoException {
    thrown.expect(LadoInvalidoException.class);
```

```

        thrown.expectMessage("lado invalido");
        Triangulo.classificaTriangulo(0, 5, 5);
    }

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

    @Test
    public void testLadoCInvalidoZero() throws
LadoInvalidoException {
        String resultado = Triangulo.classificaTriangulo(5, 5, 0);
        assertEquals("NAO FORMA TRIANGULO", resultado);
    }

    @Test
    public void
testNaoFormaTrianguloASomaDeDoisLadosMenorQueTerceiro1() throws
LadoInvalidoException {
        String resultado = Triangulo.classificaTriangulo(1, 2, 5);
        assertEquals("NAO FORMA TRIANGULO", resultado);
    }

    @Test
    public void
testNaoFormaTrianguloASomaDeDoisLadosMenorQueTerceiro2() throws
LadoInvalidoException {
        String resultado = Triangulo.classificaTriangulo(5, 1, 2);
        assertEquals("NAO FORMA TRIANGULO", resultado);
    }

    @Test
    public void
testNaoFormaTrianguloASomaDeDoisLadosMenorQueTerceiro3() throws
LadoInvalidoException {
        String resultado = Triangulo.classificaTriangulo(1, 5, 2);
        assertEquals("NAO FORMA TRIANGULO", resultado);
    }

```


```
}




@Test
public void
testNaoFormaTrianguloASomaDeDoisLadosIgualAoTerceiro() throws
LadoInvalidoException {
    String resultado = Triangulo.classificaTriangulo(1, 2, 3);
    assertEquals("ESCALENO", resultado);
}

@Test
public void testConstrutorParaCoberturaTriangulo() {
    @SuppressWarnings("unused")
    Triangulo instancia = new Triangulo();
    org.junit.Assert.assertNotNull(instancia);
}
}
```

Test Runner for Java

× Close Test Coverage

- ✓  testEquilatero() \$(symbol-cla...
- ✓  testEscaleno() \$(symbol-class...
- ✓  testIsoscelesAB() \$(symbol-cl...
- ✓  testIsoscelesAC() \$(symbol-cl...
- ✓  testIsoscelesBC() \$(symbol-cl...
- ✓  testLadoAInvalidoNegativo()...
- ✓  testLadoAInvalidoZero() \$(sy...
- ✓  testLadoBInvalidoNegativo()...
- ✓  testLadoBInvalidoZero() \$(sy...
- ✓  testLadoCInvalidoNegativo()...
- ✓  testLadoCInvalidoZero() \$(sy...
- ✓  testConstrutorParaCobertur...
- ✓  testNaoFormaTrianguloASo...
- ✓  testNaoFormaTrianguloASo...
- ✓  testNaoFormaTrianguloASo...
- ✓  testNaoFormaTrianguloASo...

 Triangulo.java	100%		%TESTS
 Word	12/12 statements covered (100.00%)		
> resour	2/2 functions covered (100.00%)		
test/jav	22/22 branches covered (100.00%)		

5)WordsTest

```
package codigos;

import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class WordsTest {

    private Words wordsInstance = new Words();

    @Test
    public void testStringNula() {
        assertEquals(-1, wordsInstance.countWords(null));
    }

    @Test
    public void testStringVazia() {
        assertEquals(0, wordsInstance.countWords(""));
    }

    @Test
    public void testStringSoEspacos() {
        assertEquals(0, wordsInstance.countWords("   "));
    }

    @Test
    public void testStringSemLetras() {
        assertEquals(0, wordsInstance.countWords("123 !@# $%^&*~.,;"));
    }

    @Test
    public void testPalavraUmCharR() {
        assertEquals(0, wordsInstance.countWords("r"));
    }

    @Test
    public void testPalavraUmCharS() {

```

```
        assertEquals(0, wordsInstance.countWords("s"));
    }

    @Test
    public void testPalavraDoisCharsR() {
        assertEquals(1, wordsInstance.countWords("ar"));
    }

    @Test
    public void testPalavraDoisCharsS() {
        assertEquals(1, wordsInstance.countWords("as"));
    }

    @Test
    public void testPalavraNaoTerminaRS() {
        assertEquals(0, wordsInstance.countWords("teste"));
    }

    @Test
    public void testPalavraValidaTerminaS() {
        assertEquals(1, wordsInstance.countWords("casas"));
    }

    @Test
    public void testPalavraValidaTerminaR() {
        assertEquals(1, wordsInstance.countWords("andar"));
    }

    @Test
    public void testExemploArPuro() {
        assertEquals(1, wordsInstance.countWords("Ar puro"));
    }

    @Test
    public void testExemploFazerValer() {
        assertEquals(2, wordsInstance.countWords("Fazer valer a
pena"));
    }

    @Test
    public void testExemploAsLetrasRes() {
```

```
        assertEquals(4, wordsInstance.countWords("As letras res são  
usadas sempre"));  
    }  
  
    @Test  
    public void testMultiplasPalavrasTerminamRS() {  
        assertEquals(2, wordsInstance.countWords("amores flores  
cantam"));  
    }  
  
    @Test  
    public void testPalavrasComMultiplosEspacos() {  
        assertEquals(2, wordsInstance.countWords("carros  
motoristas"));  
    }  
  
    @Test  
    public void testPalavraFimStringTerminaS() {  
        assertEquals(1, wordsInstance.countWords("atlas"));  
    }  
  
    @Test  
    public void testPalavraFimStringNaoTerminaRS() {  
        assertEquals(0, wordsInstance.countWords("agua"));  
    }  
  
    @Test  
    public void testPalavraFimStringTerminaR() {  
        assertEquals(1, wordsInstance.countWords("solar"));  
    }  
  
    @Test  
    public void testPalavrasCurtasRSNaoContadas() {  
        assertEquals(0, wordsInstance.countWords("a s e r"));  
    }  
  
    @Test  
    public void testPalavrasCasoMisto() {  
        assertEquals(2, wordsInstance.countWords("Sabores Cores"));  
    }  
}
```

```
@Test
public void testPalavrasComNumerosSimbolos() {
    assertEquals(1, wordsInstance.countWords("programadores1
gostam de Java!"));
}

@Test
public void testPalavraTerminarComNaoLetra() {
    assertEquals(1, wordsInstance.countWords("amor."));
}

@Test
public void testPalavraTerminasComNaoLetra() {
    assertEquals(1, wordsInstance.countWords("casas!"));
}





@Test
public void testFraseVariasPalavras() {
    assertEquals(7, wordsInstance.countWords("Os programadores
devem criar testes para validar seus algoritmos."));
}




@Test
public void testPalavraAcentoTerminaS() {
    assertEquals(1, wordsInstance.countWords("país"));
}




@Test
public void testPalavraAcentoTerminaR() {
    assertEquals(1, wordsInstance.countWords("éter"));
}
}
```

Test Runner for Java

× Close Test Coverage


- ✓  testExemploArPuro() \$(symb..
- ✓  testExemploAsLetrasRes() \$..
- ✓  testExemploFazerValer() \$(s..
- ✓  testFraseVariasPalavras() \$(s..
- ✓  testMultiplasPalavrasTermin..
- ✓  testPalavraAcentoTerminaR(..
- ✓  testPalavraAcentoTerminaS(..
- ✓  testPalavraDoisCharsR() \$(sy..
- ✓  testPalavraDoisCharsS() \$(sy..
- ✓  testPalavraFimStringNaoTer...
- ✓  testPalavraFimStringTermina..
- ✓  testPalavraFimStringTermina..
- ✓  testPalavraNaoTerminaRS()...
- ✓  testPalavraTerminaRComNa...
- ✓  testPalavraTerminaSComNao..
- ✓  testPalavraUmCharR() \$(sym..
- ✓  testPalavraUmCharS() \$(sym..
- ✓  testPalavraValidaTerminaR()..
- ✓  testPalavraValidaTerminaS()..
- ✓  testPalavrasCasoMisto() \$(sy..
- ✓  testPalavrasComMultiplosEs..
- ✓  testPalavrasComNumerosSi...
- ✓  testPalavrasCurtasRSNaoCo...
- ✓  testStringNula() \$(symbol-cla..
- ✓  testStringSemLetras() \$(sym..

- ✓  testStringSemLetras() \$(sym...
- ✓  testStringSoEspacos() \$(sym...
- ✓  testStringVazia() \$(symbol-cl...

 Words.java 100%  

> resour 18/18 statements covered (100.00%)

test/jav 2/2 functions covered (100.00%)

 Conta 18/18 branches covered (100.00%)