

Atividade 03 - Testes automatizados

Mateus Miranda de Frias - 8516837

IDE: IntelliJ;

Reporte de cobertura: JaCoCo

Casos de teste da classe Pedido (8 testes):

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

```
@Test
public void testTaxa15PeloValorCompra() {
    float taxa = pedido.calculaTaxaDesconto(false, "", 600);
    assertEquals(15, taxa, 0);
}
```

```
@Test
public void testTaxa15PeloTipoCliente() {
    float taxa = pedido.calculaTaxaDesconto(false, "ouro", 499);
    assertEquals(15, taxa, 0);
}
```

```
@Test
public void testTaxa10PeloValorCompra() {
    float taxa = pedido.calculaTaxaDesconto(false, "", 400);
    assertEquals(10, taxa, 0);
}
```

```
@Test
public void testTaxa10PeloTipoCliente() {
    float taxa = pedido.calculaTaxaDesconto(false, "prata", 399);
    assertEquals(10, taxa, 0);
}
```

```
@Test
public void testTaxa10PelaPrimeiraCompra() {
    float taxa = pedido.calculaTaxaDesconto(true, "", 399);
    assertEquals(10, taxa, 0);
}
```






```
@Test
public void testTaxa5PeloValorCompra() {
    float taxa = pedido.calculaTaxaDesconto(false, "", 200);
    assertEquals(5, taxa, 0);
}
```

```
@Test
public void testTaxa5PeloTipoCliente() {
    float taxa = pedido.calculaTaxaDesconto(false, "bronze", 199);
    assertEquals(5, taxa, 0);
}
```

Resultado da cobertura:

 SIN50222021AUTOMACAO\$PedidoTest.exec >  codigos >  Pedido

Pedido

Element	Missed Instructions	Cov.	Missed Branches	Cov.
 calculaTaxaDesconto(boolean, String, float)		100%		100%
 Pedido()		100%		n/a
Total	0 of 39	100%	0 of 14	100%

Casos de teste da classe Identifier (12 testes):

```
@Test
public void testClasse() { new Identifier(); }

@Test
public void testIDValido1() {
    Assert.assertEquals(true, Identifier.validaIdentificador("abc"));
}

@Test
public void testIDValido2() {
    Assert.assertEquals(true, Identifier.validaIdentificador("a"));
}

@Test
public void testIDValido3() {
    Assert.assertEquals(true, Identifier.validaIdentificador("abcdef"));
}

@Test
public void testIDInvalido1() {
    Assert.assertEquals(false, Identifier.validaIdentificador(""));
}

@Test
public void testIDInvalido2() {
    Assert.assertEquals(false, Identifier.validaIdentificador("abcdefg"));
}

@Test
public void testIDValido4() {
    Assert.assertEquals(true, Identifier.validaIdentificador("ppp6"));
}

@Test
public void testIDValido5() {
    Assert.assertEquals(true, Identifier.validaIdentificador("zzttgg"));
}

@Test
public void testIDValido6() {
```

```

    Assert.assertEquals(true, Identifier.validaIdentificador("a98877"));
}

@Test
public void testIDInvalido3() {
    Assert.assertEquals(false, Identifier.validaIdentificador("abcde#"));
}

@Test
public void testIDInvalido4() {
    Assert.assertEquals(false, Identifier.validaIdentificador(null));
}



@Test
public void testIDInvalido5() {
    Assert.assertEquals(false, Identifier.validaIdentificador("6ppp"));
}

```

Resultado da cobertura:

 SIN50222021AUTOMACAO\$IdentifierTest.exec >  codigos >  Identifier

Identifier

Element	Missed Instructions	Cov.	Missed Branches	Cov.
 validaIdentificador(String)	<div><div></div></div>	100%	<div><div></div></div>	100%
 Identifier()	<div><div></div></div>	100%		n/a
Total	0 of 57	100%	0 of 14	100%

Casos de teste da classe Triangulo (12 testes):

```

@Test
public void testClasse() { new Triangulo(); }

@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(5,4,4);
    assertEquals("ISOSCELES", resultado);
}

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

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

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

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

@Test
public void testNaoFormaTrianguloA() throws LadoInvalidoException{
    String resultado = Triangulo.classificaTriangulo(11,5,5);
    assertEquals("NAO FORMA TRIANGULO", resultado);
}

@Test
public void testNaoFormaTrianguloB() throws LadoInvalidoException{
    String resultado = Triangulo.classificaTriangulo(5,11,5);
    assertEquals("NAO FORMA TRIANGULO", resultado);
}

@Test
public void testNaoFormaTrianguloC() throws LadoInvalidoException{
    String resultado = Triangulo.classificaTriangulo(5,5,11);
    assertEquals("NAO FORMA TRIANGULO", resultado);
}

```

Resultado da cobertura:

SIN50222021AUTOMACAO\$TrianguloTest.exec > codigos > Triangulo

Triangulo

Element	Missed Instructions	Cov.	Missed Branches	Cov.
● classificaTriangulo(int, int, int)	<div><div></div></div>	100%	<div><div></div></div>	100%
● Triangulo()	<div><div></div></div>	100%		n/a
Total	0 of 59	100%	0 of 22	100%

Casos de teste para a classe ContaCorrente:

Análise de valor limite

Entrada	Classes Válidas	Classes Inválidas
saque	valor > 0 (v1) valor = saldo + limite (v4) valor < saldo + limite (v5) valor = saldo + limite quando saldo < 0 (v7) valor = saldo + limite quando limite = 0 (v8)	valor = 0 (v2) valor < 0 (v3) valor > saldo + limite (v6)

ID	Entrada	Saída Esperada	Valor Limite
1	valor=10, saldo=100, limite=100	90	v1
2	valor=0, saldo=100, limite=100	Valor invalido	v2
3	valor=-10, saldo=100, limite=100	Valor invalido	v3
4	valor=200, saldo=100, limite=100	-100	v4
5	valor=150, saldo=100, limite=100	-50	v5
6	valor=201, saldo=100, limite=100	Saldo insuficiente	v6
7	valor=10, saldo=-90, limite=100	-100	v7
8	valor=100, saldo=100, limite = 0	0	v8

```
@Test
public void saqueZero() throws Exception{
    //caso de teste v2
    thrown.expect(Exception.class);
    thrown.expectMessage("Valor invalido");
    ContaCorrente conta = new ContaCorrente(100,100);
    conta.saque(0);
}
```

```
}
```

```
@Test
```

```
public void testv1() throws Exception {  
    ContaCorrente conta = new ContaCorrente(100,100);  
    assertEquals(90,conta.saque(10),0);  
}
```

```
@Test
```

```
public void testv3() throws Exception {  
    thrown.expect(Exception.class);  
    thrown.expectMessage("Valor invalido");  
    ContaCorrente conta = new ContaCorrente(100,100);  
    conta.saque(-10);  
}
```

```
@Test
```

```
public void testv4() throws Exception {  
    ContaCorrente conta = new ContaCorrente(100,100);  
    assertEquals(-100,conta.saque(200),0);  
}
```

```
@Test
```

```
public void testv5() throws Exception {  
    ContaCorrente conta = new ContaCorrente(100,100);  
    assertEquals(-50,conta.saque(150),0);  
}
```

```
@Test
```

```
public void testv6() throws Exception {  
    thrown.expect(Exception.class);  
    thrown.expectMessage("Saldo Insuficiente");  
    ContaCorrente conta = new ContaCorrente(100,100);  
    conta.saque(201);  
}
```

```
@Test
```

```
public void testv7() throws Exception {  
    ContaCorrente conta = new ContaCorrente(-90,100);  
    assertEquals(-100,conta.saque(10),0);  
}
```

```
@Test
```

```
public void testv8() throws Exception {  
    ContaCorrente conta = new ContaCorrente(100,0);  
    assertEquals(0,conta.saque(100),0);  
}
```

```
@Test
```

```
public void testSetSaldo() throws Exception {  
    ContaCorrente conta = new ContaCorrente(100,100);  
    conta.setSaldo(200);  
}
```


```
@Test
```

```
public void testSetLimite() throws Exception {  
    ContaCorrente conta = new ContaCorrente(100,100);  
    conta.setLimite(200);  
}
```








```
@Test
public void testGetSaldo() throws Exception {
    ContaCorrente conta = new ContaCorrente(-90,100);
    assertEquals(-90,conta.getSaldo(),0);
}
```

```
@Test
public void testGetLimite() throws Exception {
    ContaCorrente conta = new ContaCorrente(-90,100);
    assertEquals(100,conta.getLimite(),0);
}
```

Resultado da cobertura:

 SIN50222021AUTOMACAO\$ContaCorrenteTest.exec >  [codigos](#) >  ContaCorrente

ContaCorrente

Element	Missed Instructions	Cov.	Missed Branches	Cov.
saque(float)		100%		100%
ContaCorrente(float, float)		100%		n/a
setSaldo(float)		100%		n/a
setLimite(float)		100%		n/a
getSaldo()		100%		n/a
getLimite()		100%		n/a
Total	0 of 55	100%	0 of 4	100%

Casos de teste para a classe Words:

ID	Entrada	Saída Esperada
1	null	-1
2	"um"	0
3	"rato"	0
4	"sapo"	0
5	"rato sapo"	0
6	"andar"	1
7	"rato andar"	1
8	"sapos"	1
9	"sapo sapos"	1
10	"andar sapos"	2
11	"r s"	0

12	"s r"	0
13	"t u"	0

```
@Test
public void testv1() throws Exception {
    assertEquals(-1, words.countWords(null));
}
```

```
@Test
public void testv2() throws Exception {
    assertEquals(0, words.countWords("um"));
}
```

```
@Test
public void testv3() throws Exception {
    assertEquals(0, words.countWords("rato"));
}
```

```
@Test
public void testv4() throws Exception {
    assertEquals(0, words.countWords("sapo"));
}
```

```
@Test
public void testv5() throws Exception {
    assertEquals(0, words.countWords("rato sapo"));
}
```

```
@Test
public void testv6() throws Exception {
    assertEquals(1, words.countWords("andar"));
}
```

```
@Test
public void testv7() throws Exception {
    assertEquals(1, words.countWords("rato andar"));
}
```

```
@Test
public void testv8() throws Exception {
    assertEquals(1, words.countWords("sapos"));
}
```

```
@Test
public void testv9() throws Exception {
    assertEquals(1, words.countWords("sapo sapos"));
}
```

```
@Test
public void testv10() throws Exception {
    assertEquals(2, words.countWords("andar sapos"));
}
```




```
@Test
public void testv11() throws Exception {
    assertEquals(0, words.countWords("r s"));
}
```





```
@Test
public void testv12() throws Exception {
    assertEquals(0, words.countWords("s r"));
}
```

```
@Test
public void testv13() throws Exception {
    assertEquals(0, words.countWords("t u"));
}
```

Resultado da cobertura:

 [SIN50222021AUTOMACAO\\$WordsTest.exec](#) >  [codigos](#) >  Words

Words

Element	Missed Instructions	Cov.	Missed Branches	Cov.
 countWords(String)	<div><div></div></div>	100%	<div><div></div></div>	100%
 Words()	<div><div></div></div>	100%		n/a
Total	0 of 61	100%	0 of 18	100%

Resultado final:

```
java 100% classes, 100% lines covered
└─ codigos 100% classes, 100% lines covered
   └─ ContaCorrente 100% methods, 100% lines covered
   └─ Identifier 100% methods, 100% lines covered
   └─ LadoInvalidoException 100% methods, 100% lines covered
   └─ Pedido 100% methods, 100% lines covered
   └─ Triangulo 100% methods, 100% lines covered
   └─ Words 100% methods, 100% lines covered
```