
matera
FINANCIAL. RISK. RETAIL.

# Legibilidade

## de código não é perfumaria

# Alessandra Kajihara e Ariane Izac

matera
FINANCIAL. RISK. RETAIL.

# CONTEXTO

- Piloto de Automação de Testes
- Objetivo: Evoluir a **qualidade** e **cobertura de testes**
- **Maiores Dificuldades**
  - **Legibilidade** do código
  - **Construções** complexas
  - **Propósito** do teste
  - **Asserts** que não validam o propósito ou ausência de asserts

matera

# **E o que é** legibilidade**?**


matera

# LEGIBILIDADE

- Facilidade de leitura e entendimento
- Bem estruturado/formatado
- Minimizar margem para falsos positivos
- Fácil manutenção


matera

# “Decifrar menos. Criar mais”

# CALMA! Juramos que têm código!


matera

# 1. Nome do teste

- Precisa ser
  - Claro e **auto-explicativo**
  - **Coerente** com o assert do teste

- **Padrões** normalmente utilizado no Java
  - **Camelcase**

# 2. Atenção com Métodos

- Métodos muito grandes
  - Tornam-se **complexos**
  - **Dificultam** entendimento

- **Reuso** de código
  - **Encapsulamento**

matera

```java
package test.TestesIniciais;

import static org.testng.Assert.assertEquals;

public class ContaTest {

    @Test
    public void realizaSaqueTest() {

        ContaCorrente contaTeste = new ContaCorrente("Teste Conta Corrente",
                                                     1234,
                                                     123456789,
                                                     4);

        assertEquals(contaTeste.getTaxa(), 5.00);
        assertEquals(contaTeste.getSaldo(), 0.00);

        contaTeste.fazDeposito(120.25);

        assertEquals(contaTeste.getSaldo(), 120.25);

        contaTeste.fazSaque(20.25);
    }
}
```

```java
package test.TestesIniciais;

import static org.testng.Assert.assertEquals;

public class ContaTest {

    @Test
    public void realizaSaqueTest() {

        ContaCorrente contaTeste = new ContaCorrente("Teste Conta Corrente",
                                                     1234,
                                                     123456789,
                                                     4);

        assertEquals(contaTeste.getTaxa(), 5.00);
        assertEquals(contaTeste.getSaldo(), 0.00);

        contaTeste.fazDeposito(120.25);

        assertEquals(contaTeste.getSaldo(), 120.25);

        contaTeste.fazSaque(20.25);
    }
}
```

```java
package test.TestesIniciais;

import static org.testng.Assert.assertEquals;

public class ContaTest {

    @Test
    public void realizaSaqueTest() {

        ContaCorrente contaTeste = new ContaCorrente("Teste Conta Corrente",
                                                     1234,
                                                     123456789,
                                                     4);

        assertEquals(contaTeste.getTaxa(), 5.00);
        assertEquals(contaTeste.getSaldo(), 0.00);

        contaTeste.fazDeposito(120.25);

        assertEquals(contaTeste.getSaldo(), 120.25);

        contaTeste.fazSaque(20.25);
    }
}
```

```java
package test.TestesIniciais;

import static org.testng.Assert.assertEquals;

public class ContaTest {

    @Test
    public void realizaSaqueTest() {

        ContaCorrente contaTeste = new ContaCorrente("Teste Conta Corrente",
                                                     1234,
                                                     123456789,
                                                     4);

        assertEquals(contaTeste.getTaxa(), 5.00);
        assertEquals(contaTeste.getSaldo(), 0.00);

        contaTeste.fazDeposito(120.25);

        assertEquals(contaTeste.getSaldo(), 120.25);

        contaTeste.fazSaque(20.25);
    }
}
```

```java
@Test
public void validaTaxaDeContaSalarioTest() {
    ContaCorrente contaTeste = dsl.criaConta(TipoConta.SALARIO.getCodigo());

    assertEquals(contaTeste.getTaxa(), 5.00);
}

@Test
public void validaSaldoDeContaSalarioTest() {
    ContaCorrente contaTeste = dsl.criaConta(TipoConta.SALARIO.getCodigo());

    assertEquals(contaTeste.getSaldo(), 0.00);
}

@Test
public void validaSaldoContaSalarioAposDepositoTest() {
    ContaCorrente contaTeste = dsl.criaConta(TipoConta.SALARIO.getCodigo());

    contaTeste.fazDeposito(120.25);

    assertEquals(contaTeste.getSaldo(), 120.25);
}

@Test
public void validaSaldoContaSalarioAposSaqueTest() {

    ContaCorrente contaTeste = dsl.criaContaComSaldo(120.25, TipoConta.SALARIO.getCodigo());

    contaTeste.fazSaque(20.25);

    assertEquals(contaTeste.getSaldo(), 100.00);
}
```

```
13      @Test
14      public void validaTaxaDeContaSalarioTest() {
15          ContaCorrente contaTeste = dsl.criaConta(TipoConta.SALARIO.getCodigo());
16
17          assertEquals(contaTeste.getTaxa(), 5.00);
18      }
19
20      @Test
21      public void validaSaldoDeContaSalarioTest() {
22          ContaCorrente contaTeste = dsl.criaConta(TipoConta.SALARIO.getCodigo());
23
24          assertEquals(contaTeste.getSaldo(), 0.00);
25      }
26
27      @Test
28      public void validaSaldoContaSalarioAposDepositoTest() {
29          ContaCorrente contaTeste = dsl.criaConta(TipoConta.SALARIO.getCodigo());
30
31          contaTeste.fazDeposito(120.25);
32
33          assertEquals(contaTeste.getSaldo(), 120.25);
34      }
35
36      @Test
37      public void validaSaldoContaSalarioAposSaqueTest() {
38
39          ContaCorrente contaTeste = dsl.criaContaComSaldo(120.25, TipoConta.SALARIO.getCodigo());
40
41          contaTeste.fazSaque(20.25);
42
43          assertEquals(contaTeste.getSaldo(), 100.00);
44      }
```

```
13      @Test
14      public void validaTaxaDeContaSalarioTest() {
15          ContaCorrente contaTeste = dsl.criaConta(TipoConta.SALARIO.getCodigo());
16
17          assertEquals(contaTeste.getTaxa(), 5.00);
18      }
19
20      @Test
21      public void validaSaldoDeContaSalarioTest() {
22          ContaCorrente contaTeste = dsl.criaConta(TipoConta.SALARIO.getCodigo());
23
24          assertEquals(contaTeste.getSaldo(), 0.00);
25      }
26
27      @Test
28      public void validaSaldoContaSalarioAposDepositoTest() {
29          ContaCorrente contaTeste = dsl.criaConta(TipoConta.SALARIO.getCodigo());
30
31          contaTeste.fazDeposito(120.25);
32
33          assertEquals(contaTeste.getSaldo(), 120.25);
34      }
35
36      @Test
37      public void validaSaldoContaSalarioAposSaqueTest() {
38
39          ContaCorrente contaTeste = dsl.criaContaComSaldo(120.25, TipoConta.SALARIO.getCodigo());
40
41          contaTeste.fazSaque(20.25);
42
43          assertEquals(contaTeste.getSaldo(), 100.00);
44      }
```

# 3. Uso de Enums

- Definição de domínios
- Facilita a escrita e leitura dos testes
- Enum próprio para o teste

matera

# 4. Data Driven Testing

- O que é Data Driven?
  - Testes orientados a dados
  - Uso de matrizes (tabela-verdade)
- Quando utilizar
  - Vários testes com o mesmo fluxo e dados diferentes de entrada e/ou saída
- Como?

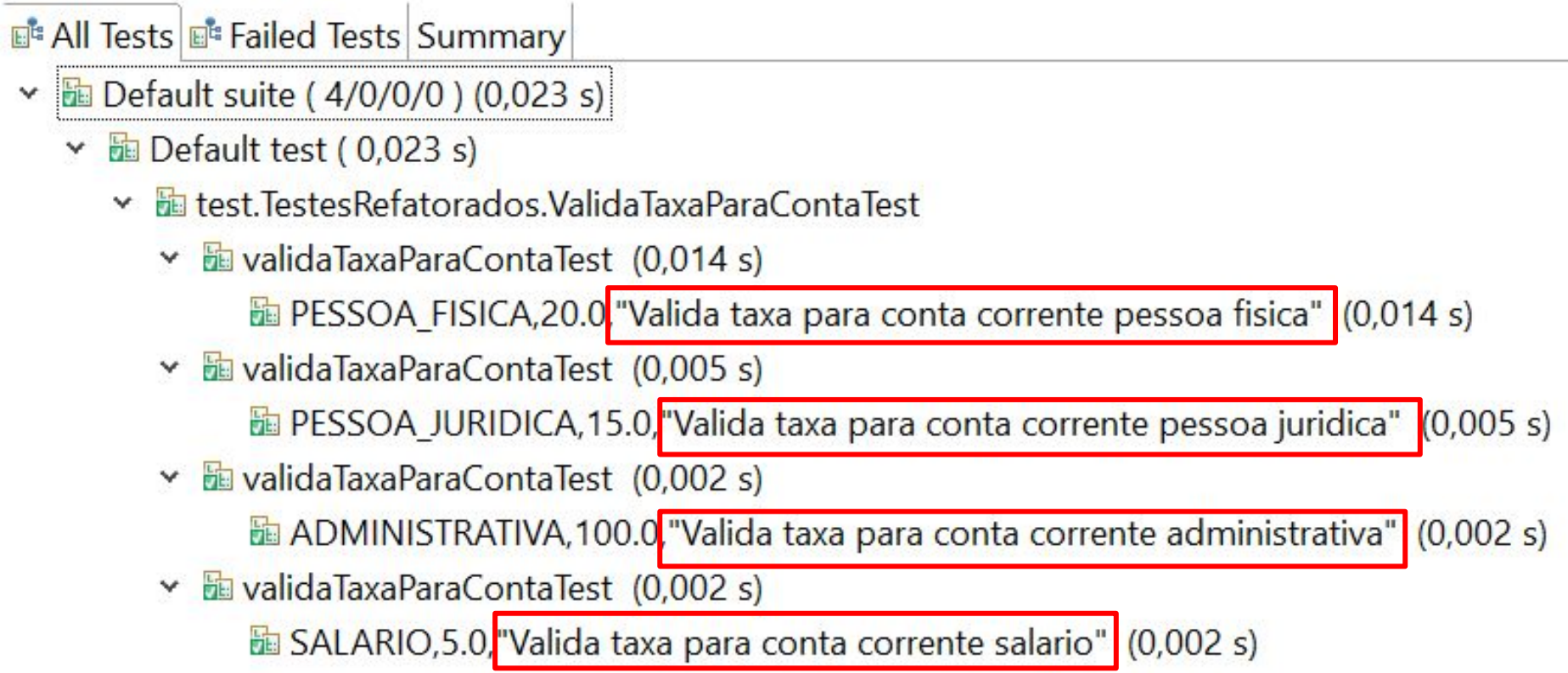
matera

```java
@Test
public void validaTaxaParaContaPessoaFisicaTest() {

    ContaCorrente contaTeste = new ContaCorrente("Fulano",
                                                 1234,
                                                 123456789,
                                                 1);


    assertEquals(contaTeste.getTaxa(), 20.00);
}

@Test
public void validaTaxaParaContaPessoaJuridicaTest() {

    ContaCorrente contaTeste = new ContaCorrente("Beltrano",
                                                 1234,
                                                 123456789,
                                                 2);


    assertEquals(contaTeste.getTaxa(), 15.00);
}
```

```java
@Test
public void validaTaxaParaContaAdministrativaTest() {

    ContaCorrente contaTeste = new ContaCorrente("Ciclano",
                                                 1234,
                                                 123456789,
                                                 3);


    assertEquals(contaTeste.getTaxa(), 100.00);
}

@Test
public void validaTaxaParaContaSalarioTest() {

    ContaCorrente contaTeste = new ContaCorrente("Alguém",
                                                 1234,
                                                 123456789,
                                                 4);


    assertEquals(contaTeste.getTaxa(), 5.00);
}
```

```java
@DataProvider(name = "massaDeDados")
public Object[][] criacaoMassaDados() {
    return new Object[][] {
        // Tipo Conta                    Taxa      Cenario
        {  TipoConta.PESSOA_FISICA    ,   20.00  ,  "Valida taxa para conta corrente pessoa física"  },
        {  TipoConta.PESSOA_JURIDICA  ,   15.00  ,  "Valida taxa para conta corrente pessoa jurídica"},
        {  TipoConta.ADMINISTRATIVA   ,   100.00 ,  "Valida taxa para conta corrente administrativa" },
        {  TipoConta.SALARIO          ,   5.00   ,  "Valida taxa para conta corrente salario"        }
    };
}

@Test (dataProvider = "massaDeDados")
public void validaTaxaParaContaTest( TipoConta tipoConta,
                                     double taxa,
                                     String cenario) {

    ContaCorrente contaTeste = new ContaCorrente("Fulano Beltrano Ciclano",
                                                 1234,
                                                 1234567890,
                                                 tipoConta.getCodigo());

    assertEquals(contaTeste.getTaxa(), taxa);
}
```

```java
@DataProvider(name = "massaDeDados")
public Object[][] criacaoMassaDados() {
    return new Object[][] {
        // Tipo Conta                    Taxa      Cenario
        {  TipoConta.PESSOA_FISICA   ,   20.00  ,  "Valida taxa para conta corrente pessoa física"  },
        {  TipoConta.PESSOA_JURIDICA ,   15.00  ,  "Valida taxa para conta corrente pessoa jurídica"},
        {  TipoConta.ADMINISTRATIVA  ,   100.00 ,  "Valida taxa para conta corrente administrativa" },
        {  TipoConta.SALARIO         ,   5.00   ,  "Valida taxa para conta corrente salario"        }
    };
}

@Test (dataProvider = "massaDeDados")
public void validaTaxaParaContaTest( TipoConta tipoConta,
                                     double taxa,
                                     String cenario) {

    ContaCorrente contaTeste = new ContaCorrente("Fulano Beltrano Ciclano",
                                                 1234,
                                                 1234567890,
                                                 tipoConta.getCodigo());

    assertEquals(contaTeste.getTaxa(), taxa);
}
```


All Tests
Failed Tests
Summary
Default suite ( 4/0/0/0 ) (0,023 s)
Default test ( 0,023 s)
test.TestesRefatorados.ValidaTaxaParaContaTest
validaTaxaParaContaTest (0,014 s)
PESSOA_FISICA,20.0,"Valida taxa para conta corrente pessoa fisica" (0,014 s)
validaTaxaParaContaTest (0,005 s)
PESSOA_JURIDICA,15.0,"Valida taxa para conta corrente pessoa juridica" (0,005 s)
validaTaxaParaContaTest (0,002 s)
ADMINISTRATIVA,100.0,"Valida taxa para conta corrente administrativa" (0,002 s)
validaTaxaParaContaTest (0,002 s)
SALARIO,5.0,"Valida taxa para conta corrente salario" (0,002 s)

# 5. Uso de DSL

- **O que é DSL?**
  - **D**omain **S**pecific **L**anguage: Linguagem de domínio específico
  - Pequenas Linguagens  - Resolve um **problema específico**
  - Camada de **abstração**

- **Tipos de DSL**
  - **DSL Interna:** Utiliza linguagem host
  - **DSL Externa:** Utiliza uma outra\nova linguagem para atender o domínio

# 5. Uso de DSL

- **Quando utilizar?**
  - Casos **complexos**
  - Casos com muito reuso de código
- **Como utilizar?**
  - Linguagem de **negócio**
  - **Encapsulamento** do método
- **Vantagens**
  - Facilita o entendimento do código - **Intuitivo**
  - Aumenta a  produtividade - **manutenção**

matera

```java
@Test
public void validaTaxaParaContaPessoaFisicaTest() {

    ContaCorrente contaTeste = new ContaCorrente("Fulano",
                                                 1234,
                                                 123456789,
                                                 1);

    assertEquals(contaTeste.getTaxa(), 20.00);
}

@Test
public void validaTaxaParaContaPessoaJuridicaTest() {

    ContaCorrente contaTeste = new ContaCorrente("Beltrano",
                                                 1234,
                                                 123456789,
                                                 2);

    assertEquals(contaTeste.getTaxa(), 15.00);
}
```

```java
@Test
public void validaTaxaParaContaAdministrativaTest() {

    ContaCorrente contaTeste = new ContaCorrente("Ciclano",
                                                 1234,
                                                 123456789,
                                                 3);

    assertEquals(contaTeste.getTaxa(), 100.00);
}

@Test
public void validaTaxaParaContaSalarioTest() {

    ContaCorrente contaTeste = new ContaCorrente("Alguém",
                                                 1234,
                                                 123456789,
                                                 4);

    assertEquals(contaTeste.getTaxa(), 5.00);
}
```

```java
public class ValidaTaxaParaContaTest {

    ContaCorrenteDsl dsl = ContaCorrenteDsl.instanciaDsl();

    @DataProvider(name = "massaDeDados")
    public Object[][] criacaoMassaDados() {
        return new Object[][] {
            // Tipo Conta                    Taxa      Cenario
            {  TipoConta.PESSOA_FISICA    ,   20.00  ,  "Valida taxa para conta corrente pessoa física"  },
            {  TipoConta.PESSOA_JURIDICA  ,   15.00  ,  "Valida taxa para conta corrente pessoa jurídica"},
            {  TipoConta.ADMINISTRATIVA   ,   100.00 ,  "Valida taxa para conta corrente administrativa" },
            {  TipoConta.SALARIO          ,   5.00   ,  "Valida taxa para conta corrente salario"        }
        };
    }

    @Test (dataProvider = "massaDeDados")
    public void validaTaxaParaContaTest( TipoConta tipoConta,
                                         double taxa,
                                         String cenario) {


        ContaCorrente contaTeste = dsl.criaConta(tipoConta.getCodigo());

        assertEquals(contaTeste.getTaxa(), taxa);
    }
}
```

# Estrutura DSL

```
5  public class ContaCorrenteDsl {
6
7      private ContaCorrenteDsl() {
8
9      }
10
11     public static ContaCorrenteDsl instanciaDsl() {
12         return new ContaCorrenteDsl();
13     }
14
15     public ContaCorrente criaConta(int tipo) {
16         return new ContaCorrente("Teste Conta Corrente",
17                                  1234,
18                                  123456789,
19                                  tipo);
20     }
21
22     public ContaCorrente criaContaComSaldo(double valorSaldo, int tipo) {
23         ContaCorrente conta = new ContaCorrente("Teste Conta Corrente",
24                                                 1234,
25                                                 123456789,
26                                                 tipo);
27
28         conta.fazDeposito(valorSaldo);
29
30         return conta;
31     }
32
33 }
```

# E pra fechar...


- Legibilidade não é perfumaria
- Facilita **entendimento**
- Facilita a **manutenção**
- Deixa seu **código limpo**
- Adotar **padrões** ou **diretrizes**
    - Linguagem
    - Time (Empresa)

matera

**“Test code** *is just as* **important** *as* **production code...”**

Robert C. Martin - Clean Code

# Alessandra Kajihara

**Analista de Testes**
Há 7 anos

**Matera Systems**
Há 5 anos

## CONTATOS

Linkedin: **Alessandra Kajihara** Email: **sahkaji@gmail.com**

matera
FINANCIAL. RISK. RETAIL.

# Ariane Izac

**Analista de Testes**
Há 12 anos

**Matera Systems**
Há 7 anos
Blogueira

**Grupo no LinkedIN**
Diário de uma Paixão:
Teste de Software

## CONTATOS

Linkedin: **Ariane Izac** Email: **afizac@gmail.com** Twitter: **@arianizac**

matera
FINANCIAL. RISK. RETAIL.

# REFERÊNCIAS


**[1]** https://www.guj.com.br/t/legibilidade-de-codigo/61896/6

**[2]** https://medium.com/equals-lab/tem-um-tempinho-para-conversarmos-sobre-boas-pr%C3%A1ticas-de-programa%C3%A7%C3%A3o-com-java-51c5bcc220ea

**[3]** https://pt.slideshare.net/FelipeVolpone/legibilidade-do-cdigo

**[4]** https://www.alura.com.br/curso-online-refatorando-na-pratica-com-java

**[5]** https://google.github.io/styleguide/javaguide.html


matera

# REFERÊNCIAS


**[6]** https://www.infoq.com/br/presentations/arquiteturas-profissionais-javaee

**[7]** https://www.martinfowler.com/bliki/DslQandA.html

**[8]** https://www.devmedia.com.br/convencoes-de-codigo-java/23871

**[9]** http://www.eliasnogueira.com/parametrizacao-de-dados-para-automacao-de-teste/


matera

LET'S **CREATE** THE **FUTURE**