



Aula 2: Ferramenta JUnit

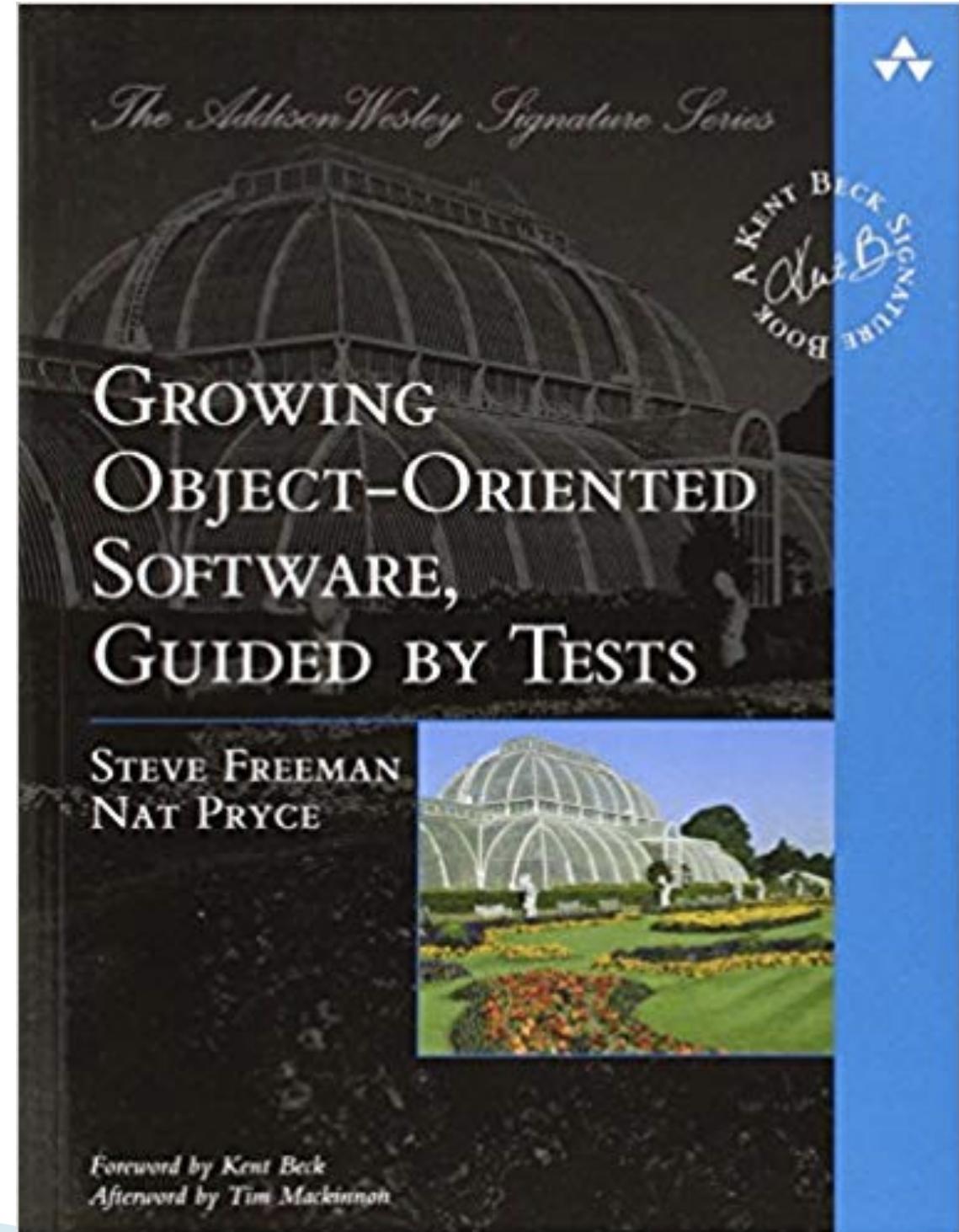
Prof. Thiago

Departamento de Informática e Matemática
Aplicada - DIMAp

Referências:

- ▶ Artigo:
 - Test Infected: Programmers Love Writing Tests
<http://junit.sourceforge.net/doc/testinfected/testing.htm>
- ▶ Tutorial:
 - How to write hard to test code?
<http://misko.hevery.com/2009/10/28/how-to-write-hard-to-test-code-what-to-look-for-when-reviewing-other-peoples-hard-to-test-code/>
- ▶

Referências:



Objetivos da aula

- ▶ Discutir a importância dos testes de unidade
- ▶ Entender o framework JUnit
- ▶ Construir e executar testes de unidade
- ▶ Discutir sobre o que dificulta os Testes de Unidade

Agenda

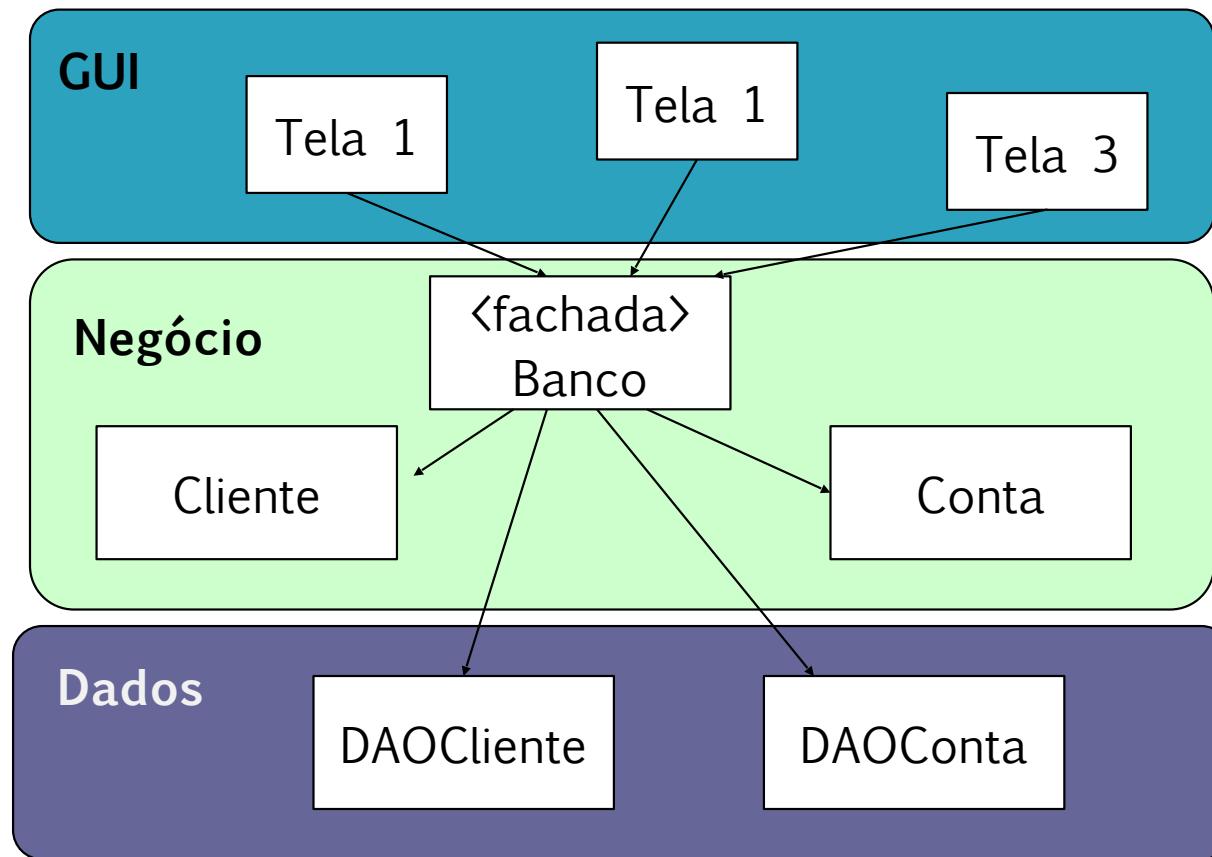
- Motivação
- O framework JUnit
- Passo a passo...
- Diferenças JUnit 3, JUnit4, JUnits
- Mão na Massa

Agenda

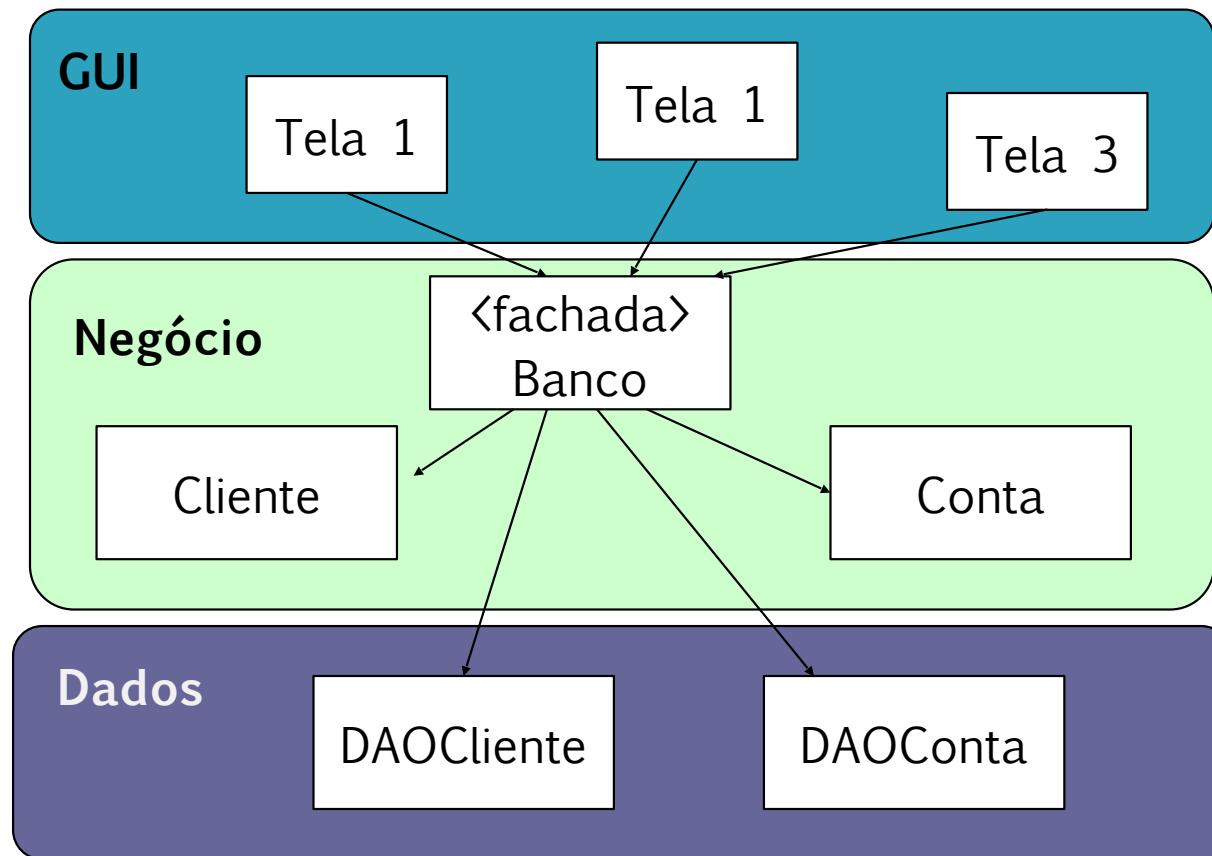
- Motivação
- O framework JUnit
- Passo a passo...
- Diferenças JUnit 3, JUnit4, JUnits
- Mão na Massa

QualitiInternetBanking

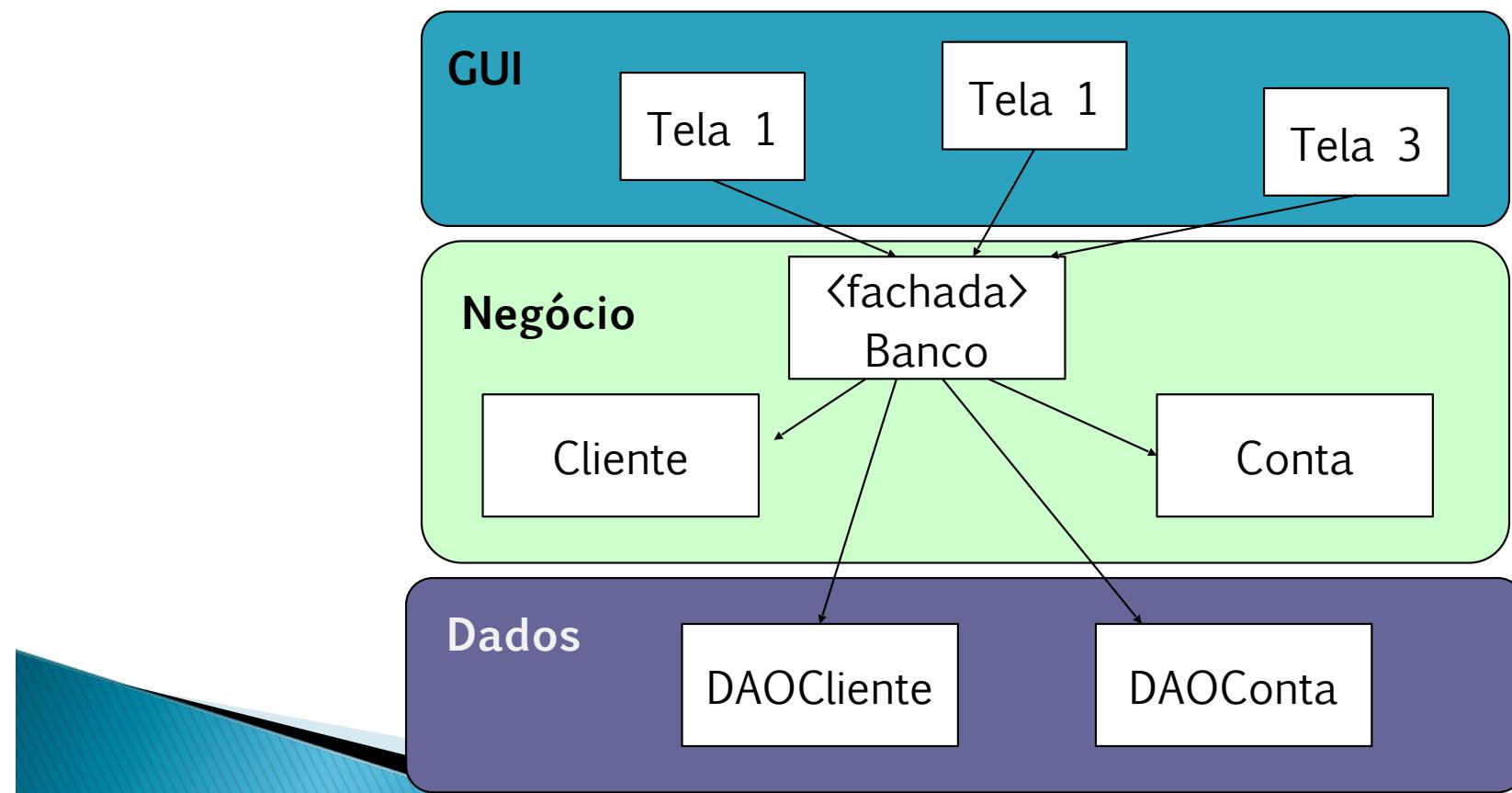
Obs: Um sistema que não tem testes



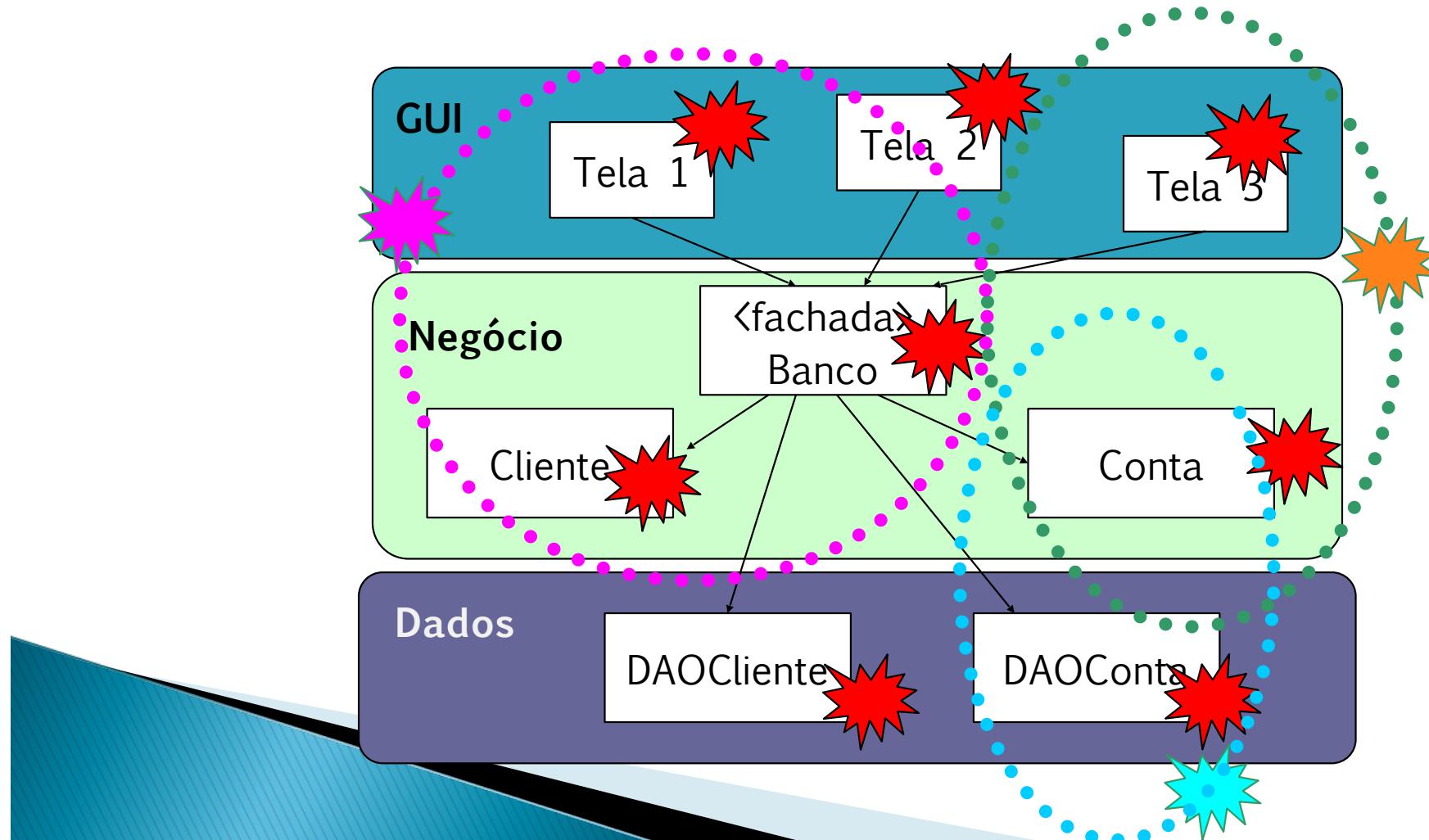
Imagine que...
na release 4.0 a Transferência
entre Contas deixa de funcionar...



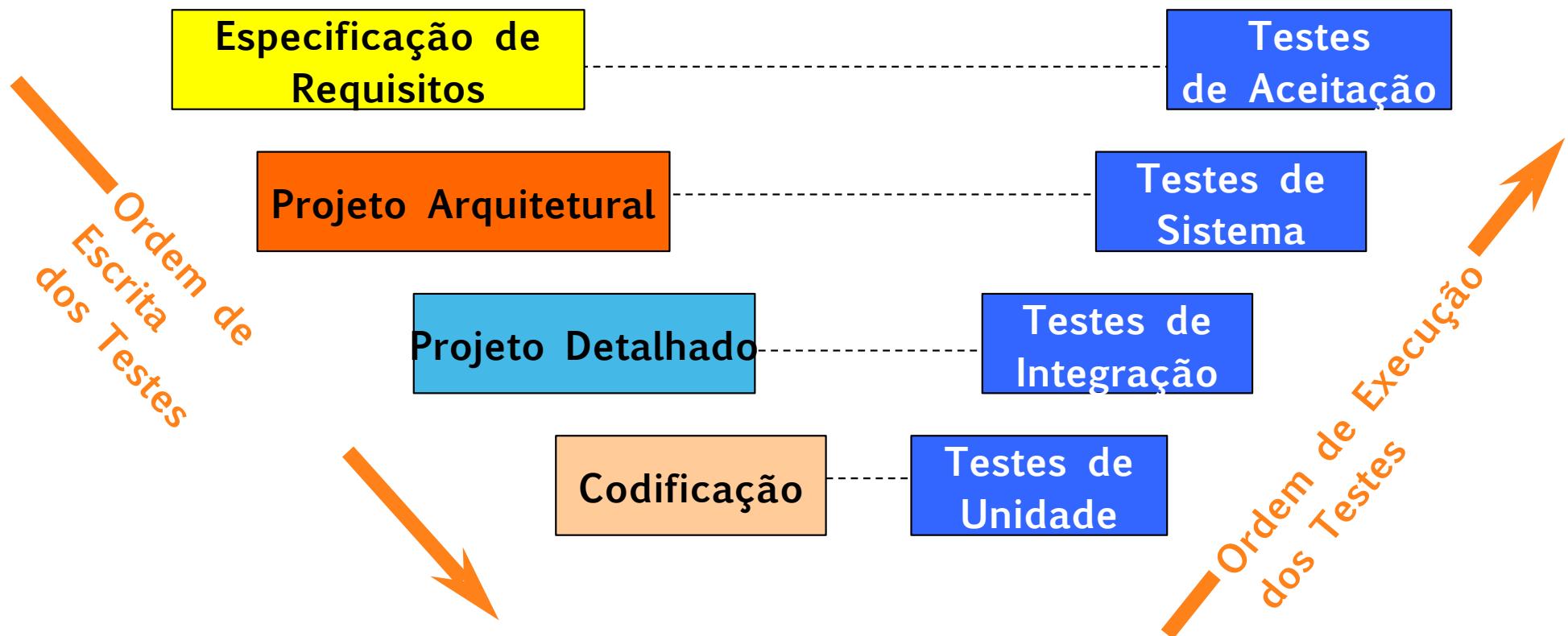
Onde pode estar o Bug??



Onde pode estar o Bug??



Estágios de Teste



Testes de Sistema

- **Vantagem:**
 - Ganhamos confiança nos “happy paths” do sistema
- **Desvantagem**
 - Custosos.
 - Necessário Debug

Teste de Integração

- **Vantagem:**

- Ganhamos confiança nos subsistemas
- Testes focam na interação entre as classes

- **Desvantagem**

- É mais fácil de simular cenários de falha mas...
- ... ainda precisamos de debug

Teste de Unidade

- **Vantagem:**

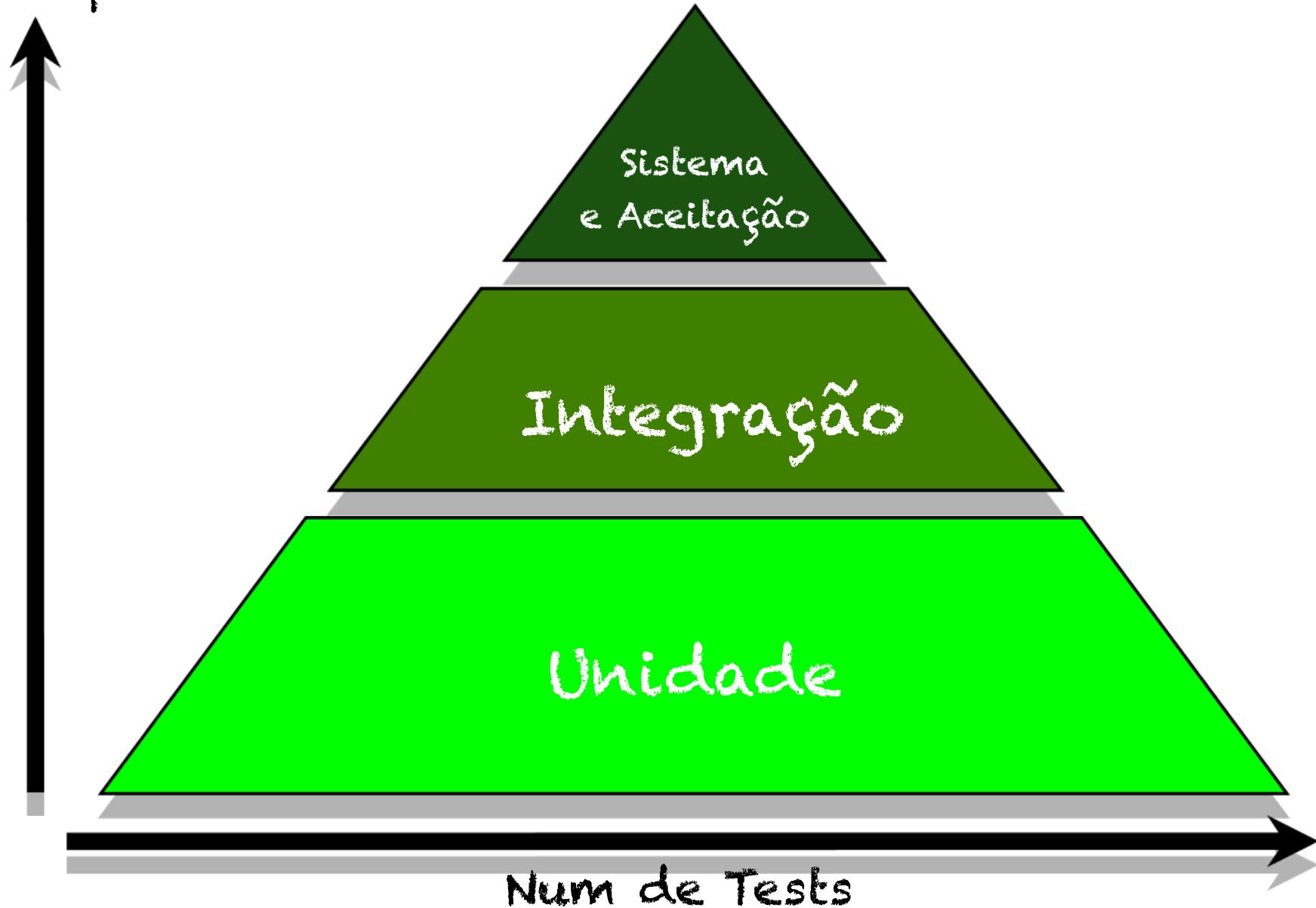
- Testa classes isoladamente
- Pode simular condições de erro
- Executa mais rápido que os testes anteriores

- **Desvantagem**

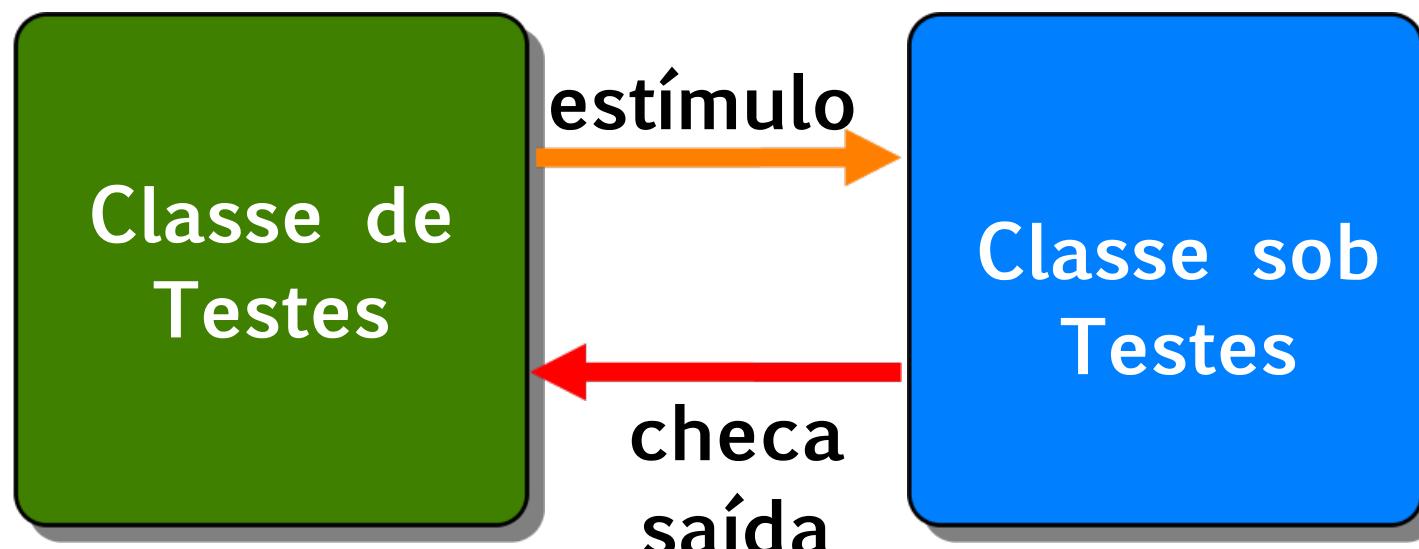
- Testa partes do sistema isoladamente
- Bugs de integração e sistema não são detectados

Testes: Desempenho x Quantidade

Tempo de exec.



Motivação para Teste de Unidade



Quiz:

Como você testaria o
método debitar
(sem usar o JUnit)

```
public class Conta{  
    ...  
    public Conta (String p_id, double p_saldo) {  
        this.saldo = p_saldo;  
        this.codigo = p_id;  
    }  
    public double getSaldo() {...}  
  
    public void debitar(double valor) throws  
        OperacaoIllegalException {  
        if(valor > 0 && saldo >= valor){  
            saldo = saldo-valor;  
        }else{  
            throw new OperacaoIllegalException();  
        }  
    }  
}
```

Resposta 1

```
public class TesteConta{  
    ...  
    public void testDebitar() {  
        Conta c = new Conta ("123", 10.0);  
        try{  
            c.debitar(5.0);  
            if ( c.getSaldo(5.0) == 5.0 ){  
                System.out.println("CORRETO");  
            }else {  
                System.out.println("ERRO");  
            }  
        }catch(OperacaoIlegalException ope){  
            System.out.println("ERRO");  
        }  
    }  
}
```

Resposta 1 (cont.)

```
public class TesteConta{  
    ...  
    public static void main (String args[] ) {  
        TesteConta ct = new TesteConta () ;  
        ct.testDebitar () ;  
    }  
}
```

Quais são as limitações
desta solução?



Quiz

Imagine se criarmos vários métodos de testes.. deste jeito...

```
public static void main (String args[] ) {  
    TesteConta ct = new TesteConta () ;  
    ct.testDebitar(5.0) ;  
    ct.testDebitarNegativo(-1.0) ;  
    ct.testCreditar(5.0) ;  
    ct.testCreditarNegativo(-1.0) ;  
    ...  
}
```

Quiz

O log vai ficar mais ou menos assim:

CORRETO

CORRETO

CORRETO

CORRETO

ERRO

...

Quiz

Olhar a saída dos testes
no log, dificulta a vida
de quem testa...



Quiz

Se um teste lançar uma exceção, vai impedir que os demais testes sejam executados...



Agenda

- Motivação
- O framework JUnit
- Passo a passo...
- Diferenças JUnit 3, JUnit4, JUnits
- Mão na Massa

Agenda

- Motivação
- O framework JUnit
-
- Passo a passo...
- Diferenças JUnit 3, JUnit4, JUnits
- Mão na Massa

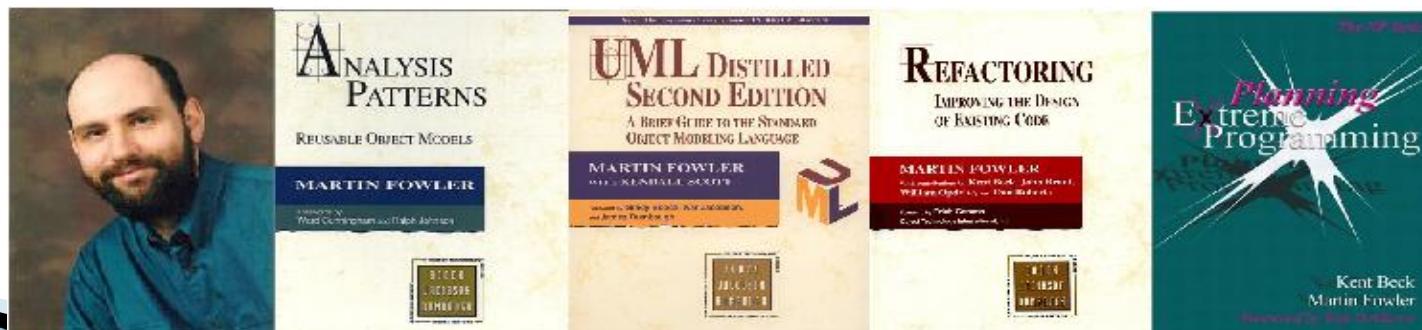
JUnit

“Never in the field of software development was so much owed by so many to so few lines of code.”

Martin Fowler

“Nunca o desenvolvimento de software deveu tanto a tão poucas linhas de código.”

Martin Fowler



Framework JUnit

Dá suporte a **definição** e
execução de testes de unidade de
programas Java.

5 JUnit 5

JUnit 4

The new major version of the programmer-friendly testing framework for Java

User Guide

Javadoc

Code & Issues

Q & A

Support JUnit

About

JUnit 5 is the next generation of JUnit. The goal is to create an up-to-date foundation for developer-side testing on the JVM. This includes focusing on Java 8 and above, as well as enabling many different styles of testing.

JUnit 5 is the result of [JUnit Lambda](#) and its [crowdfunding campaign on Indiegogo](#).

Resources

Upcoming Events

2018-10-08

Spock vs JUnit 5 - Clash of the Titans at [JDD](#) in Cracow, Poland
Marcin Zajęczkowski

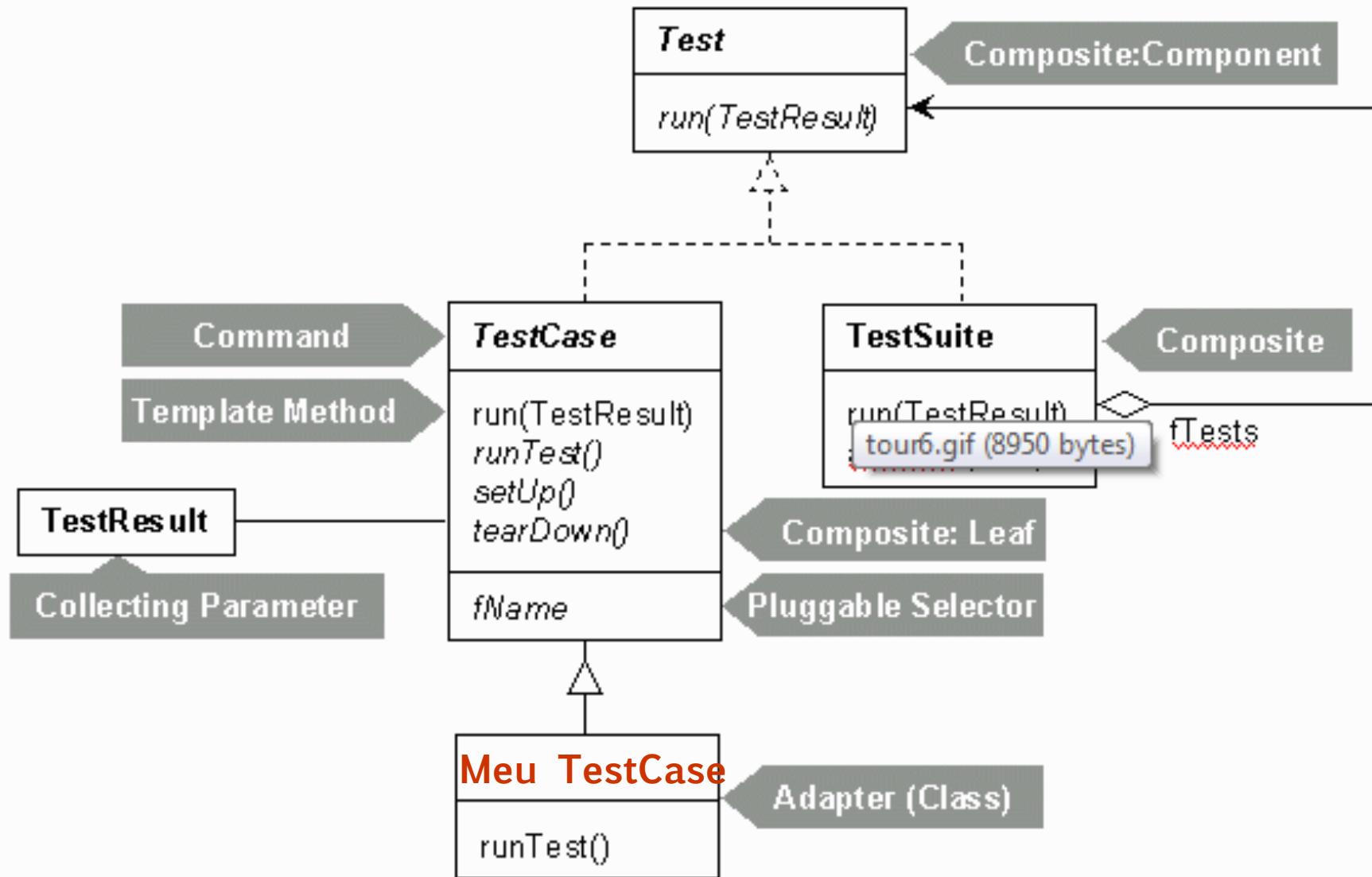
2018-11-06

JUnit 5: Platform & Jupiter API at [JUG Bonn](#) in Bonn,

Framework JUnit

- ▶ Desenvolvido por Erich Gamma (*Design Patterns*) and Kent Beck (XP).
- ▶ Um excelente exemplo da aplicação de vários *Padrões de Projeto*.

Arquitetura



JUnit

- ▶ Idéia básica:

- Para testar uma classe, criamos uma classe de testes correspondente.

Integração com IDEs: Eclipse, NetBeans...

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** workspace - Teste de Software - A Senha/src/test/testCE/testPinoCE.java - Eclipse IDE
- Toolbar:** Standard Eclipse toolbar with icons for file operations, search, and navigation.
- ActionBar:** Shows the current project (TestSiteGoogle), package-info.java, jpacman-framewo, ClienteDAOTest, and testPinoCE.
- Left Panel (Outline View):** Shows the test results:
 - Finished after 0.046 seconds
 - Runs: 2/2 Errors: 0 Failures: 0
 - test.testCE.testPinoCE [Runner: JUnit 4] (0.001 s)
 - testInvalido (0.001 s) (selected)
 - testValido (0.000 s)
- Right Panel (Editor):** Displays the Java code for testPinoCE.java:

```
1 package test.testCE;
2
3 import static org.junit.Assert.assertEquals;
4
5
6 public class testPinoCE {
7
8     Pino pino;
9
10    @Before
11    public void setUp() throws Exception {
12        int numero = new Random().nextInt(7);
13        pino = new Pino(numero);
14    }
15
16    @After
17    public void tearDown() throws Exception {
18    }
19
20    @Test
21    public void testValido() {
22        assertTrue(pino.getCor() <= 6 && pino.getCor() >= 0);
23    }
24
25    @Test
26    public void testInvalido() {
27        pino = new Pino(20);
28        assertEquals("Inválido", Cor.getCorasString(pino.getCor()));
29    }
30
31}
```
- Bottom Left:** Failure Trace button.

Integração com IDEs: Eclipse, NetBeans...

The screenshot shows the Eclipse IDE interface with the following details:

- Toolbar:** Standard Eclipse toolbar with various icons for file operations, search, and navigation.
- ActionBar:** Shows the current workspace path: "workspace - ExerciciosJUnit/src_test/banco/ContaTest.java - Eclipse IDE".
- Left Panel (JUnit View):**
 - Shows the message "Finished after 0.034 seconds".
 - Summary of test results: "Runs: 1/1", "Errors: 0", "Failures: 1".
 - A blue bar highlights the failed test: "testDebitarSucesso [Runner: JUnit 4] (0.001 s)".
- Code Editor:** Displays the Java code for "ContaTest.java".

```
1 package banco;
2
3 //só para os testes com @Test
4 import static org.junit.Assert.*;
5
6 public class ContaTest extends TestCase {
7
8     public void testDebitarSucesso() throws OperacaoIlegalException{
9         Conta c = new Conta("123", 20);
10        c.debitar(10);
11        assertEquals(15,c.getSaldo(),0.0);
12    }
13}
```
- Bottom Left (Failure Trace):**
 - Shows the error message: "junit.framework.AssertionFailedError: expected:<15.0> but was:<10.0>".
 - Stack trace: "at banco.ContaTest.testDebitarSucesso(ContaTest.java:14)".

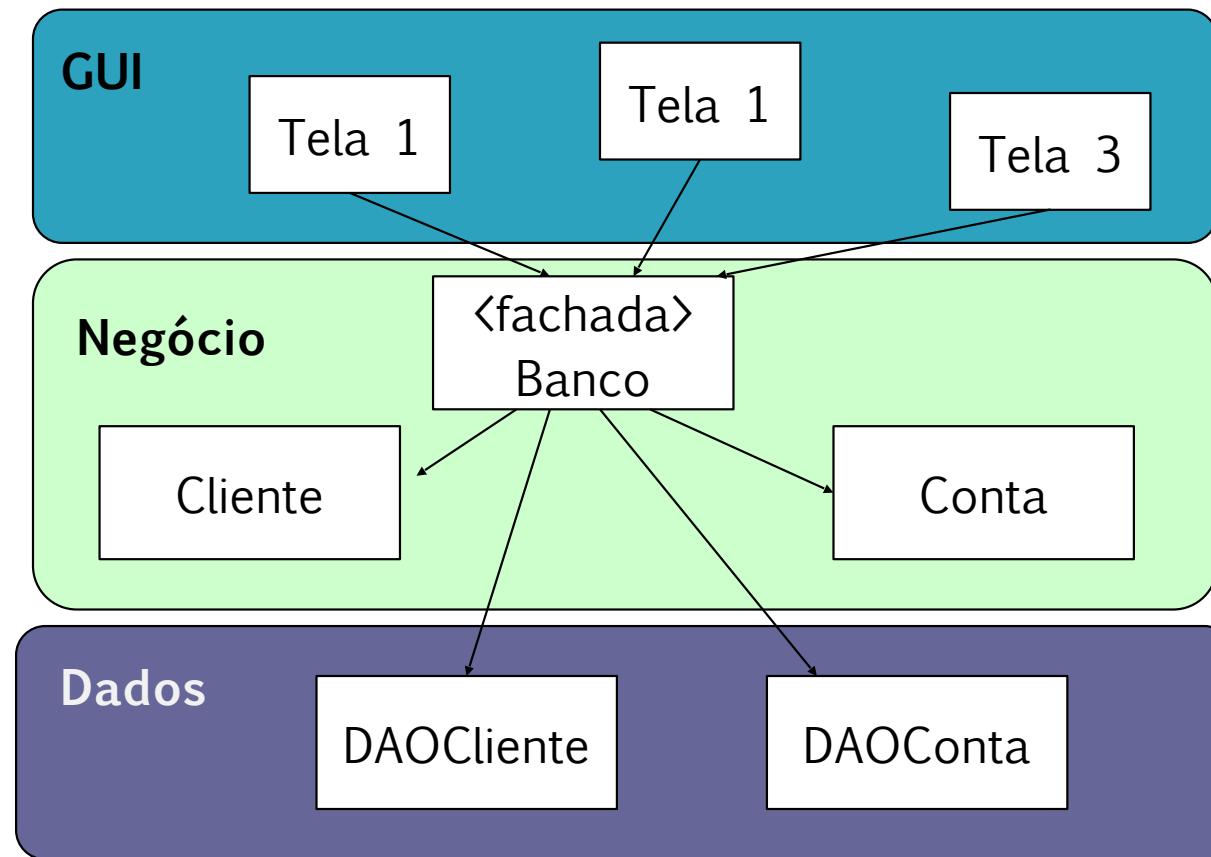
Agenda

- Motivação
- O framework JUnit
- Passo a passo...
- Diferenças JUnit 3, JUnit4, JUnits
- Mão na Massa

Agenda

- Motivação
- O framework JUnit
- Passo a passo...
- Diferenças JUnit 3, JUnit4, JUnits
- Mão na Massa

Construindo Testes de Unidade para o QualitiInternetBanking



Passo 1: Selecionar a classe a ser testada.

Test Driven Development:

Implementar o teste de Unidade
ser antes da classe a ser testada.

Passo 1 (cont.)

```
public class Conta{  
    double saldo;  
    String codigo;  
  
    public Conta (String p_id, double p_saldo) {  
        this.saldo = p_saldo;  
        this.codigo = p_id;  
    }  
  
    public void debitar(double valor) throws  
        OperacaoIllegalException {  
        if(valor > 0 && saldo >= valor) {  
            saldo = saldo-valor;  
        }else{  
            throw new OperacaoIllegalException();  
        }  
    }  
}
```

Passo 1 (cont.)

```
...
public void creditar(double valor) throws
    OperacaoIllegalException{
    if(valor > 0){
        saldo = saldo+valor;
    }else{
        throw new OperacaoIllegalException();
    }
}

//Retorna true se o saldo e o numero da conta
//forem os mesmos e false caso contrario.
public boolean equals (Conta c2) {
...
}
```

Passo 2: Construir a classe de testes

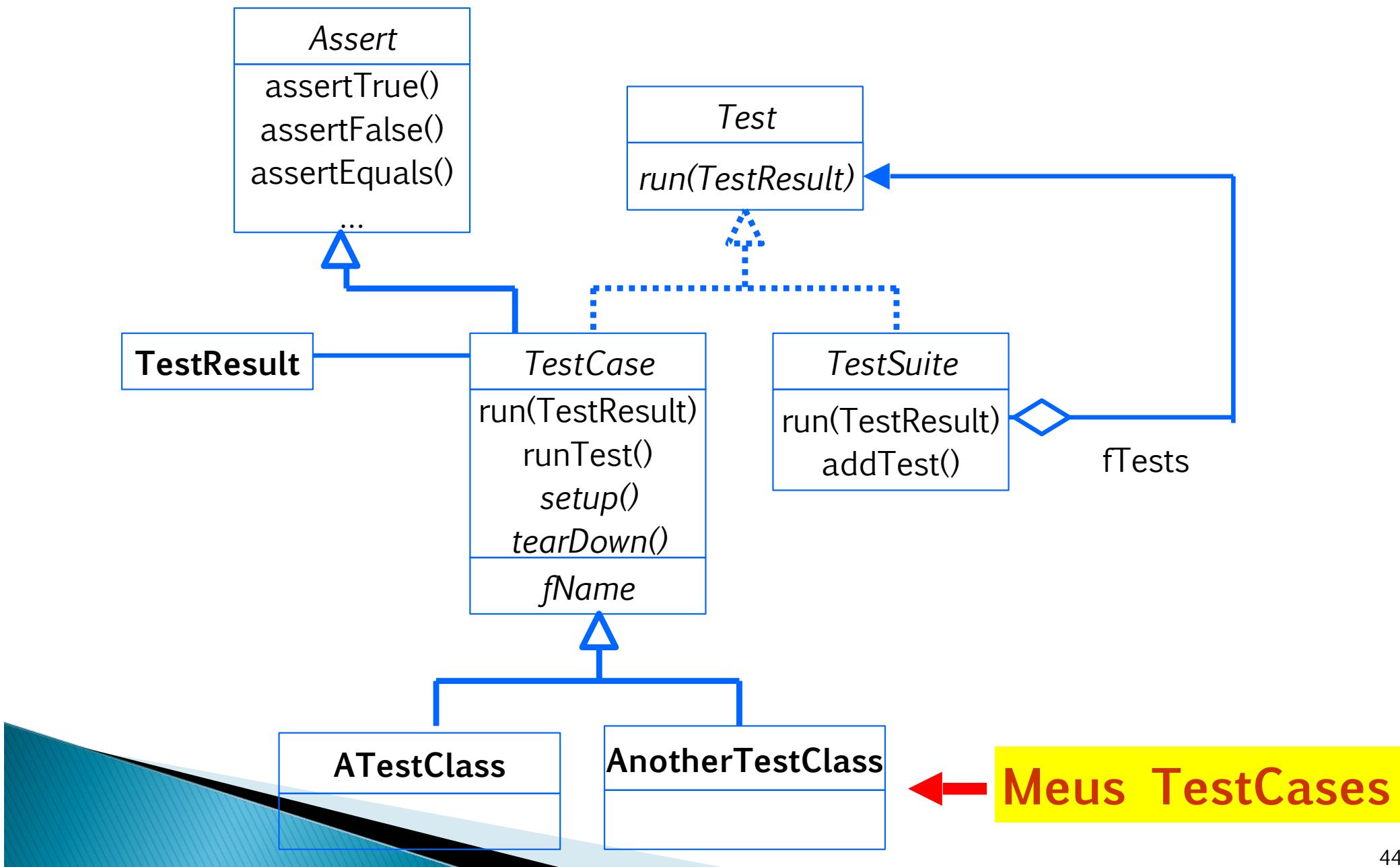
- ▶ Herda da classe TestCase do framework JUnit. (**Versão 3.x**)
- ▶ Padrão de nomenclatura sugerido:
 - <Nome da Classe> + Test = ContaTest

Passo 2: Construir a classe de testes

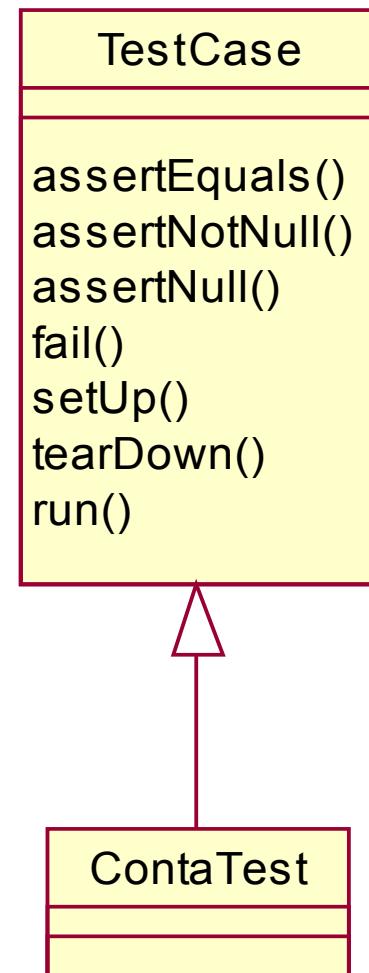
- ▶ Herda da classe TestCase do framework JUnit. (**Versão 3.x**)
- ▶ Padrão de nomenclatura sugerido:
 - <Nome da Classe> + Test = ContaTest

Ferramentas XUnit (PHP, C++) se assemelham a versão 3.

Passo 2: Construir a classe de testes



Passo 2: Construir a classe de testes



Passo 3: Criar os métodos de teste.

- ▶ Restrição até a versão 3.x:
 - devem iniciar com a palavra “test” e
 - possuir tipo de retorno void.
- ▶ Devem testar condições de **sucesso** e **falha**.

Passo 3.1: Testando cenários de sucesso

```
1. public class ContaTest extends TestCase {  
2.     public void testSaqueBemSucedido () {  
3.         Conta c1 = new Conta("123", 50.0);  
4.         c1.debitar(49.0);  
5.         assertEquals(c1.getSaldo(), 1.0);  
6.     }  
7. }
```

Passo 3.2: Testando cenários de falha - Lançamento de Exceções

```
public void testSaqueUltrapassaLimite() {  
  
    Conta c1 = new Conta ("123", 50.0);  
    c1.debitar(70.0);  
  
}  
...  
}
```

Este teste está correto?
Vamos executar...



Package Explorer Hierarchy JUnit X

Finished after 0,047 seconds

Runs: 2/2 Errors: 1 Failures: 0

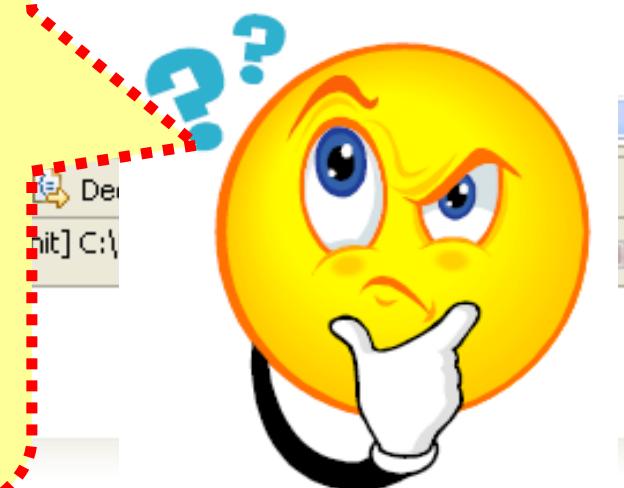
br.ufrn.sinfo.ContaTest [Runner: JUnit 3] (0,000 s)

- testSaqueBemSucedido (0,000 s)
- testSaqueUltrapassaLimite (0,000 s)

Conta c1 = new Conta("123");
c1.debitar(49.0);
assertEquals(c1.getSaldo(), 50.0);
}

public void testSaqueUltrapassaLimite() {
 Conta c1 = new Conta("123");
 c1.debitar(70.0);
 assertEquals(c1.getSaldo(), 30.0);
}

Mas se a exceção foi lançada o comportamento do método esta correto. O teste só deve falhar quando um defeito é encontrado.



Possíveis Resultados



- *Sucesso*



- *Falha*



- *exceção ou Error*

Possíveis Resultados

- ***Sucesso***

Algum dos métodos
assert lançou
AssertionFailException

- ***Falha***

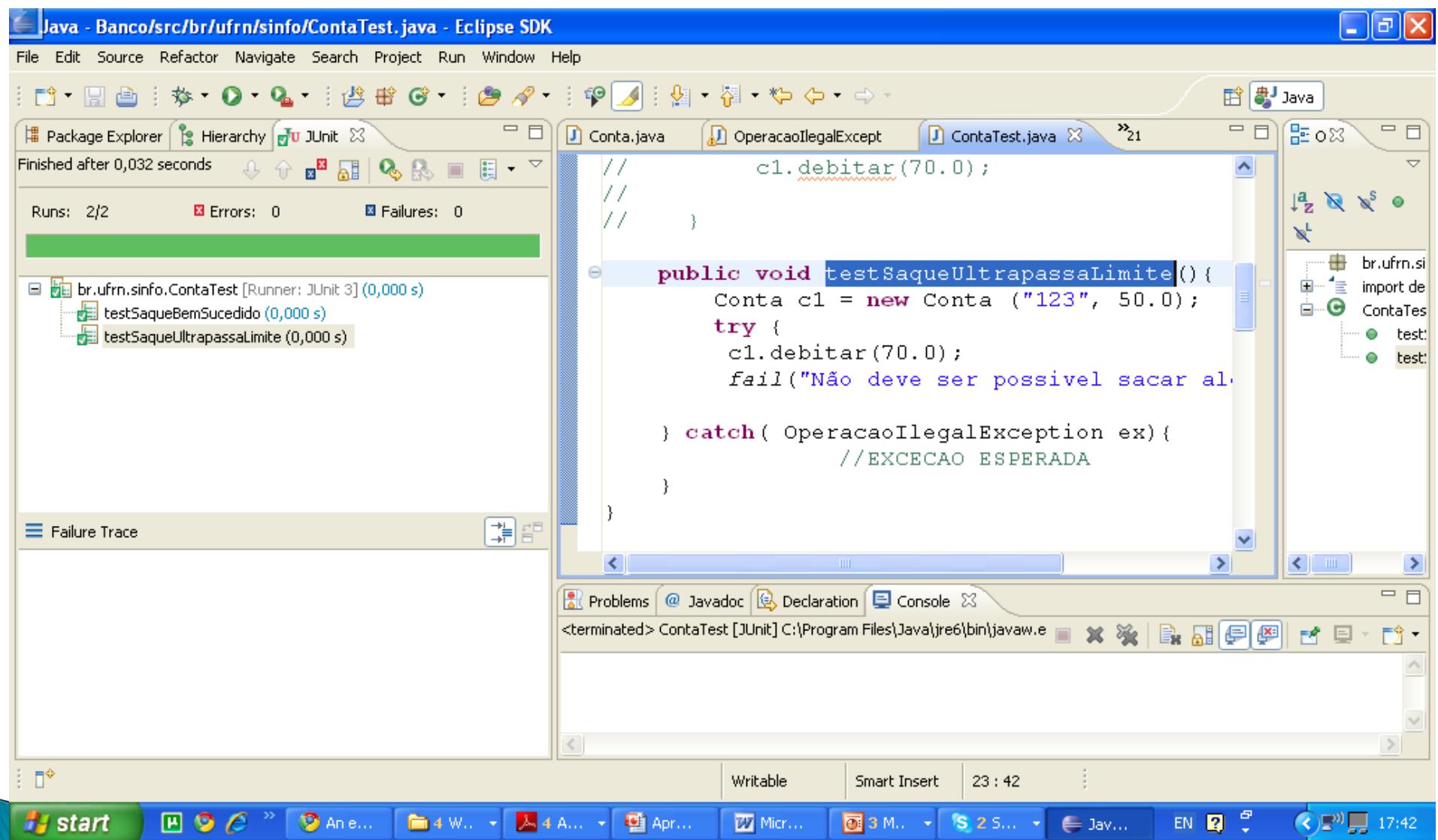
Uma exceção que não foi
lançada pelo JUnit
escapou!!

- ***exceção ou Error***

Versão correta do teste

```
12.    public void testSaqueUltrapassaLimite() {  
13.        Conta c1 = new Conta ("123", 50.0);  
14.        try {  
16.            c1.debitar(70.0);  
17.            fail("Não deve ser possivel sacar  
18.                além do saldo!");  
19.        } catch( OperacaoIllegalException ex) {  
20.            //EXCECAO ESPERADA  
21.        }  
22.    }  
23.  
24.    ...  
25.}
```

Nova execução



Passo 4: Organizar as configurações comuns aos métodos de teste.

- **setUp():**
 - chamado automaticamente antes de cada método de testes. (alocação de recursos, inicialização de objetos)
- **tearDown():**
 - chamado automaticamente ao final de cada método de testes (liberação de recursos)

Passo 4: Organizar as configurações comuns aos métodos de teste.

```
public class ContaTest extends TestCase {  
  
    Conta c1;  
  
    public void setUp() {  
        c1 = new Conta("123", 50.0);  
    }  
    ...  
}
```

Agenda

- Motivação
- O framework JUnit
- Passo a passo...
- Diferenças JUnit 3, JUnit4, JUnits
- Mão na Massa

Agenda

- Motivação
- O framework JUnit
- Passo a passo...
- Diferenças JUnit 3, JUnit4, JUnits
-
- Mão na Massa
-

June 4

JUnit4 – Principais Diferenças

- ▶ Necessita: Java 5 ou superior
- ▶ Anotações no lugar de nomes padrão.
 - ▶ **@Test** → “testXXX”
 - ▶ **@Before** → **setUp()**
 - ▶ **@After** → **tearDown()**

JUnit4 – Principais Diferenças

- ▶ JUnit 4 pode executar testes em JUnit 3
- ▶ Classe de testes **não herda de TestCase**
- ▶ `import static org.junit.Assert.*` substitui herança

Teste na versão JUnit 4.x

```
import static org.junit.Assert.*;  
  
public class ContaTestNovo {  
  
    @BeforeClass  
    public static void setUpClass() throws Exception {  
    }  
  
    @AfterClass  
    public static void tearDownClass() throws Exception {  
    }
```

Mesmo Teste na versão JUnit 4.8.2

```
...
```

```
@Before
```

```
public void setUp() throws Exception {  
}
```

```
@After
```

```
public void tearDown() throws Exception {  
}
```

```
@Test
```

```
public void testDebitar() {
```

```
    ...
```

```
}
```

```
...
```

```
}
```

Testando o lançamento de exceções

```
@Test (expected=OperacaoIlegalException.class)
public void testSaqueUltrapassaLimite(){

    Conta c1 = new Conta ("123", 50.0);
    c1.debitar(70.0);

}
```

Testando o lançamento de exceções

```
@Test(timeout=500)
public void testarCredito() {
    ...
}

@Ignore @Test
public void getSaldoTest() {
}
```

Mesmas Assertivas

```
@Test  
public void exemplosAsserts() {  
    ...  
    //Verifica que uma condição é verdadeira.  
    assertTrue(result)  
  
    //Verifica que uma condição é falsa.  
    assertFalse(result)  
  
    //Verifica que uma referência não é null.  
    assertNotNull(result)  
  
    //Verifica que uma referência é null.  
    assertNull (result)
```

Mesmas Assertivas

```
public void testarCretido() {  
    ...  
    //Verifica que duas referências apontam  
    para o mesmo objecto.  
    assertSame(esperado, result)  
  
    //Verifica que duas referências não apontam para  
    o mesmo objecto.  
    assertNotSame(esperado, result)  
  
    //Verifica que os conteúdos de dois arrays são  
    iguais.  
    assertArrayEquals(arEsperado, arResult)  
  
    //Verifica se dois objetos são iguais – equals()  
    assertEquals(esperado, result)
```

Criando uma Suite

```
import org.junit.runner.RunWith;  
import org.junit.runners.Suite;  
  
@RunWith(Suite.class)  
@Suite.SuiteClasses({ ClasseTeste1.class,  
ClasseTeste2.class })  
public class AllTests {  
  
    /* Classe vazia apenas para agrupar  
    todos os testes */  
  
}
```

Executando uma Suite

```
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class JunitRunner {

    public static void main(String[] args) {

        Result result =
        JUnitCore.runClasses(AllTests.class);

        for (Failure fail : result.getFailures())
            System.out.println(fail.toString());

        if(result.wasSuccessful())
            System.out.println("Todos os testes OK.");
    }
}
```

Executando uma Suite

- ▶ Com o org.junit.runner.JUnitCore podemos executar testes JUnit a partir de uma classe Java.
- ▶ O método runClasses() recebe uma classe de teste como argumento e retorna um valor do tipo Result.
- ▶ O valor retornado contém uma lista de resultados que permite determinar o sucesso/insucesso de cada teste.



Mão na Massa

Mão na Massa 1

- Remova os comentários de forma incremental e analise e corrija os testes da classe JUnitBasicoTest.