Teste de Software
Manual e JUnit

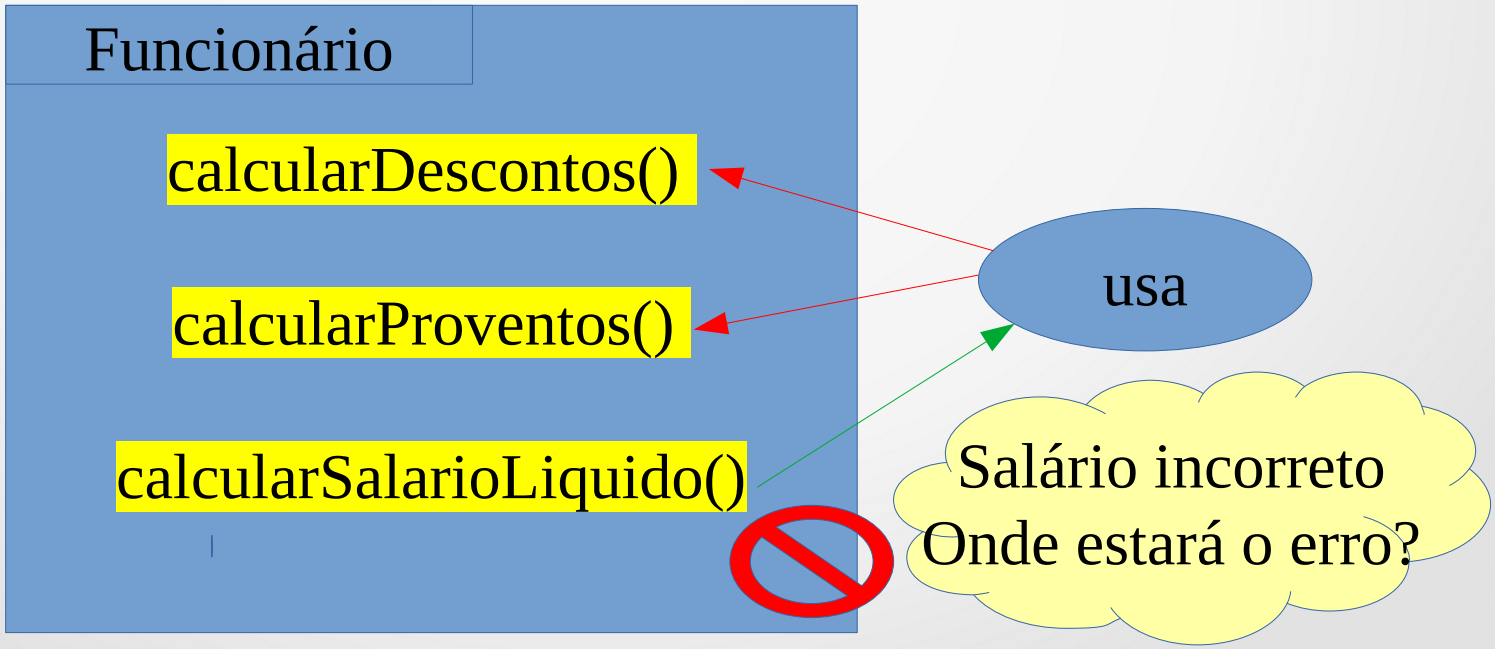
Prof. Dr. Bruno Queiroz Pinto

Testes de unidade

- Testes de unidade garantem que cada método(função) testado está produzindo o resultado esperado. Essa garantia da uma segurança maior ao programador, que poderá mudar a implementação sem medo.

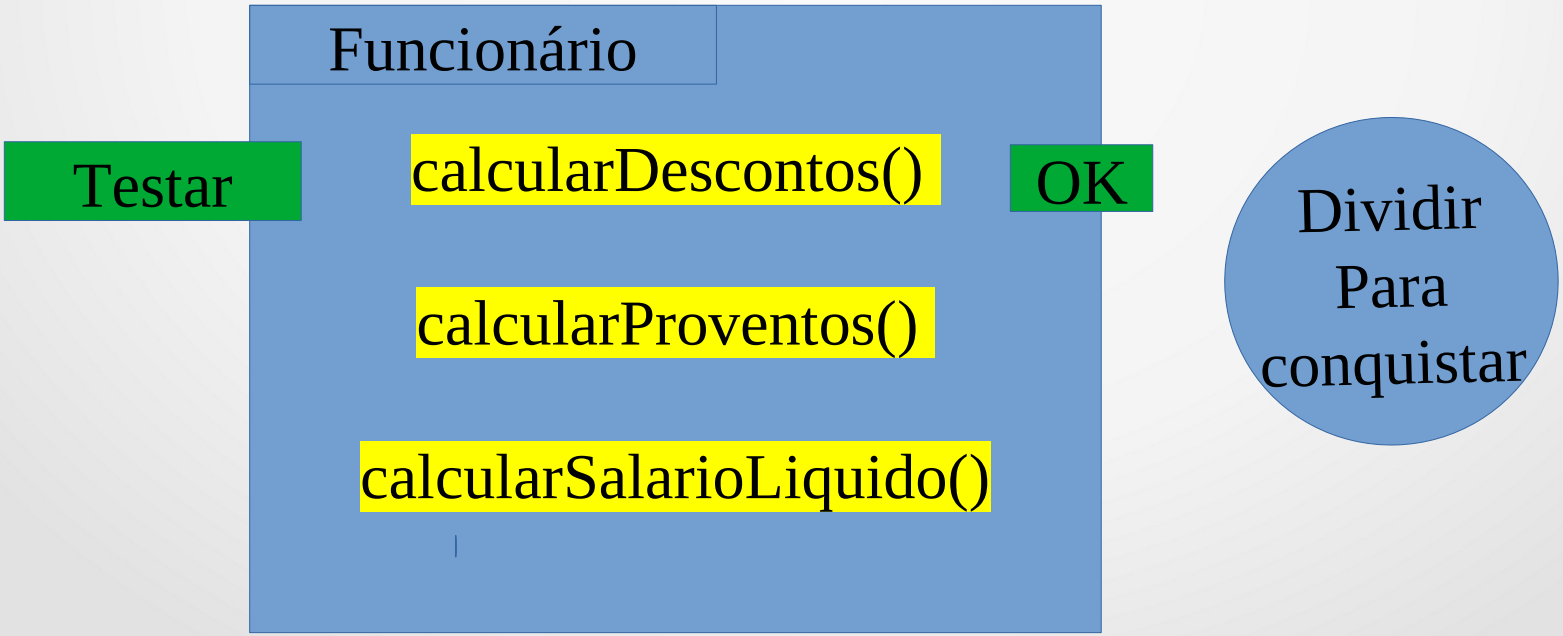
Conceito de testes unitários

- Teste capaz de analisar uma unidade de trabalho, que a IEEE define como: “Atividade capaz de testar unidades de hardware ou software ou grupo de unidades relacionadas”.



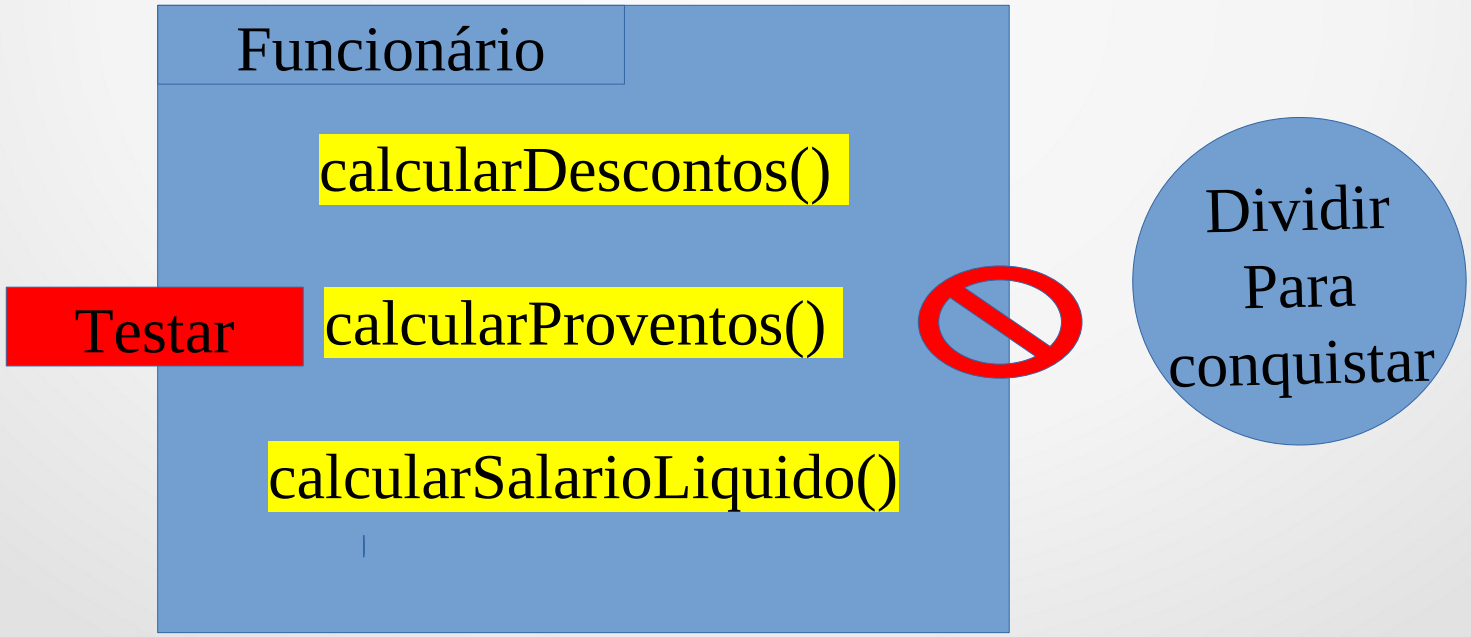
Conceito de testes unitários

- Teste capaz de analisar uma unidade de trabalho, que a IEEE define como: “Atividade capaz de testar unidades de hardware ou software ou grupo de unidades relacionadas”.



Conceito de testes unitários

- Teste capaz de analisar uma unidade de trabalho, que a IEEE define como: “Atividade capaz de testar unidades de hardware ou software ou grupo de unidades relacionadas”.



Conceito de testes unitários

- Analogia: Se você testou as tomadas individualmente, encontrará problemas mais facilmente.

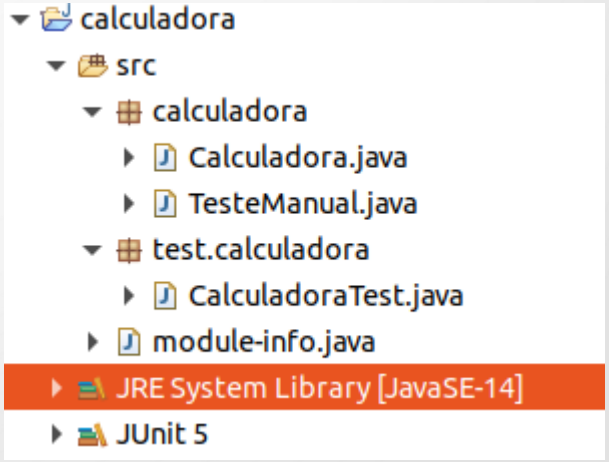
Unidade

Unidade

Testes não automatizados (sem JUnit)

- Estudo de caso:
  - um simples aplicativo de calculadora que pode somar, subtrair, multiplicar e dividir dois números inteiros.
  - Esta aplicação terá a organização de pastas conforme mostra a Figura abaixo.

Java:
- Criar um projeto no Eclipse
- Criar pacotes
- Criar classes



Código da classe Calculadora

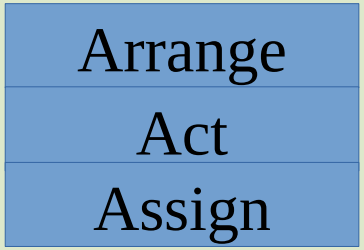
```
package calculadora;

public class Calculadora{
          public int somar(int num1, int num2) {
                    return num1 + num2;
          }

          public int subtrair(int num1, int num2) {
                    return num1 + num2;
          }

          public int multiplicar(int num1, int num2) {
                    return num1 + num2;
          }

          public int dividir(int num1, int num2) {
                    if (num2 == 0)
                              return -1;
                    return num1 / num2;
          }
}
```

Códigos com erro

Estrutura de Testes de Unidade

Estrutura de Teste

- A estrutura de Teste:
  - monta o cenário,
  - executa a ação
  - e valida a saída.



Teste de Unidade

Cenário de Teste nº 1	
Entradas	**Inteiro = 10** • **Inteiro = 20**
Saídas esperadas	**Inteiro = 30**
Descrição = Teste para validar uma entrada comum do método de soma...	
Resultado: ??	

Teste Manual

```
package calculadora;

public class TesteManual {
        public static void main (String args[]){
            Calculadora calc = new Calculadora();
            //cenário de teste
            int numero1 = 10;
            int numero2 = 20;
            int resultadoEsperado = 30;
            //executa
            int resultado = calc.somar(numero1,numero2);
            //valida resultado
            if (resultado!=resultadoEsperado){
              System.out.println("Oops! Deu um resultado não esperado: "+resultado);
            }
            else System.out.println("OK! Passou do teste.");
        }
}
```

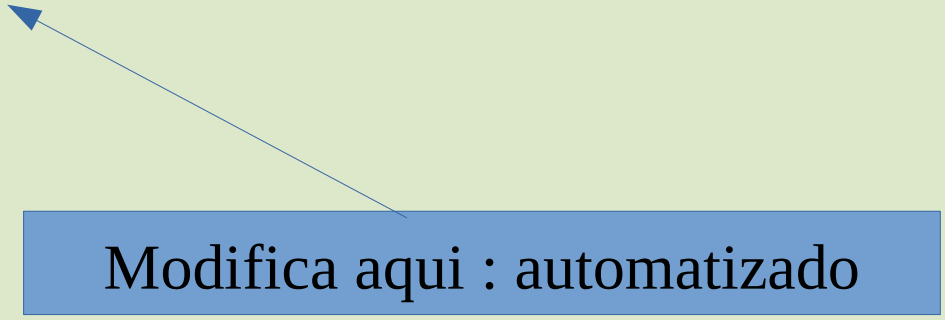
O.O Instanciar um objeto

O.O Chamar métodos

Testes **Automatizados** em Java

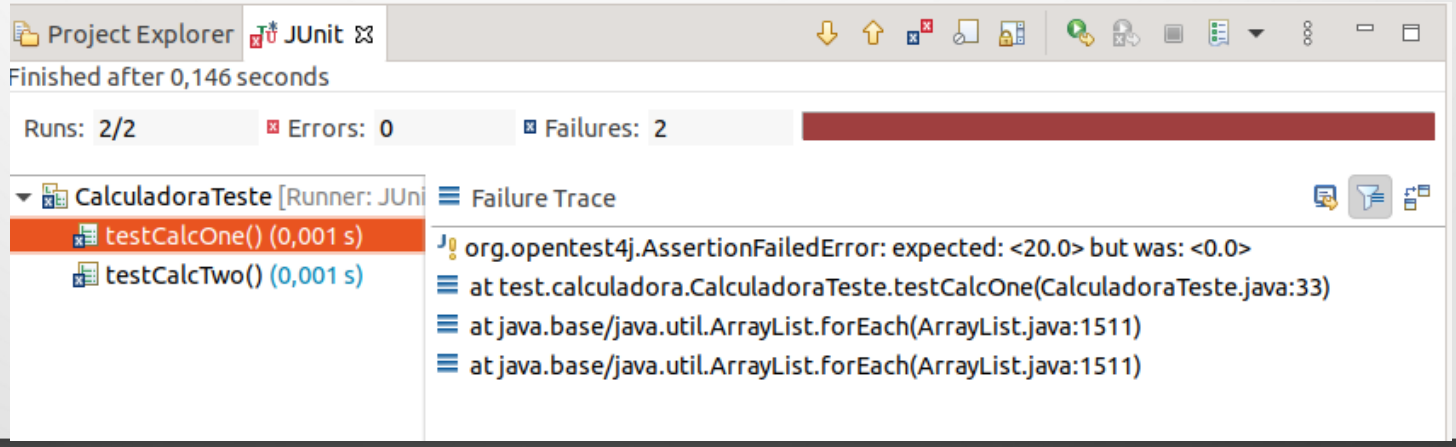
Estrutura de Teste

- A estrutura de Teste não modifica:
  - monta o cenário,
  - executa a ação
  - e valida a saída.



Testes Automatizados com JUnit

- JUnit, o mais popular framework de testes de unidade para Java.
- O framework apresenta quais casos de testes falharam e qual foi a saída incorreta produzida pelo método.
- O JUnit pinta uma barra de verde quando tudo deu certo, ou de vermelho quando algum teste falhou.
- O JUnit é tão popular que já vem incorporado nas IDE java.



Testes Automatizados com JUnit

Nosso código anterior está muito perto de ser entendido pelo JUnit. Precisamos fazer apenas algumas mudanças:

1. Um método de teste deve sempre ser público, de instância (isto é, não pode ser static) e não receber nenhum parâmetro;
2. Deve ser anotado com @Test .
3. Deve ter um nome que indica o que será testado.
   ex.:
   deveSomarDoisInteiros
   ou
   testarSomaDoisInteiros

Testes Automatizados com JUnit

Automático

```
class CalculadoraTest {
        @Test
        public void testarSomaDoisInteiros(){

            Calculadora calc = new Calculadora();
            int numero1 = 10;
            int numero2 = 20;
            int resultadoEsperado = 30;

            int resultado = calc.somar(numero1, numero2);

            // comparando a saída com o esperado
            Assertions.assertEquals(resultadoEsperado, resultado);
        }
}
```

Cenário Teste

Ação/Execução Teste

Avaliação

JUnit 5

Valor esperado

Valor obtido

Testes Automatizados com JUnit

Repare algumas mudanças no código:

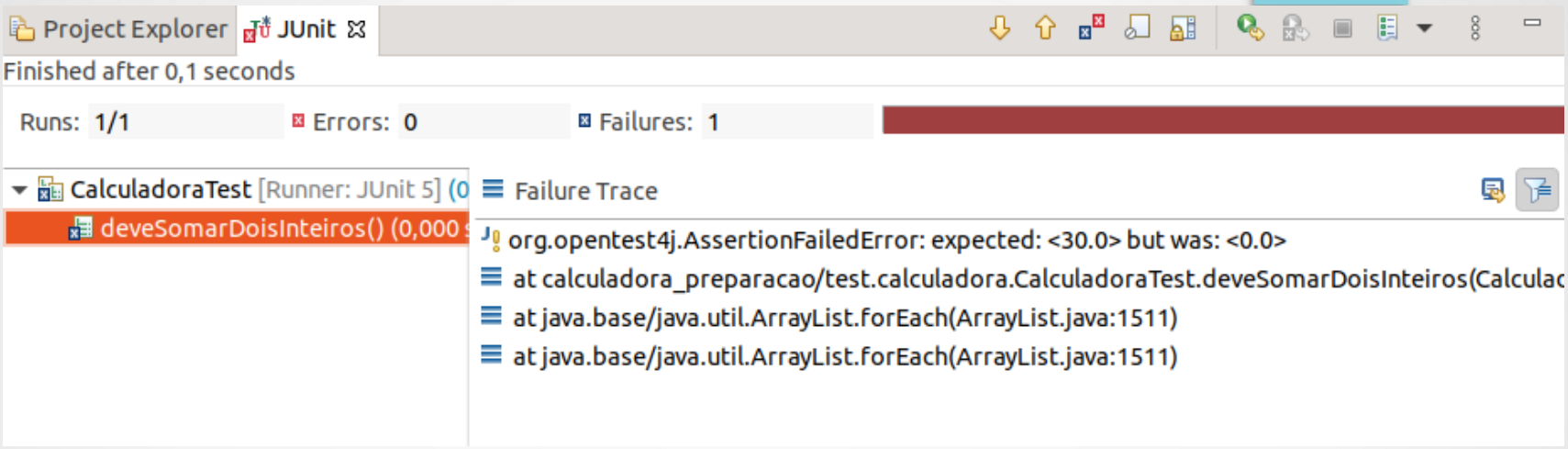
- criar um package próprio para armazenar os testes.

- nome da classe: CalculadoraTest .
  - Convenção : NomeDaClasseSobTesteTest.

- nome dos métodos de teste : testarSomaDoisInteiros.
  - nomes relacionados ao caso de teste. Conseguimos descobrir mais facilmente o que está errado em nosso sistema.

- Validar saída .
  - Utilizar instruções Asserts. assertEquals() verifica se a saída esperada é igual ao resultado gerado. (manter essa ordem)

Testes Automatizados com JUnit



- Necessidade de mudança no código, baseado no erro identificado.
- Qual mudança será necessária?

Criar novos casos de teste

- Crie novos casos de teste, um para cada método da classe Calculadora.
  - Para o método dividir, crie dois testes:
    - 1) dois números inteiros maiores que 0.
    - 2) o número divisor com valor igual a 0. (considere que o método dividir retorne o valor -1 nesse caso)
      - Depois iremos ver como testar exceptions.

- Teste a classe. Alguns testes indicarão erro no código.