Introdução ao Teste Unitário com JUnit

Agenda

- Testes de Unidade
- Junit
- Primeiro Teste
- Executando testes com Junit e NetBeans
- Passos para criação de testes unitários

Testes de Unidade

- Testes de unidade são **testes que testam apenas uma classe ou método**, verificando se seu comportamento está de acordo com o desejado.

Testes de Unidade

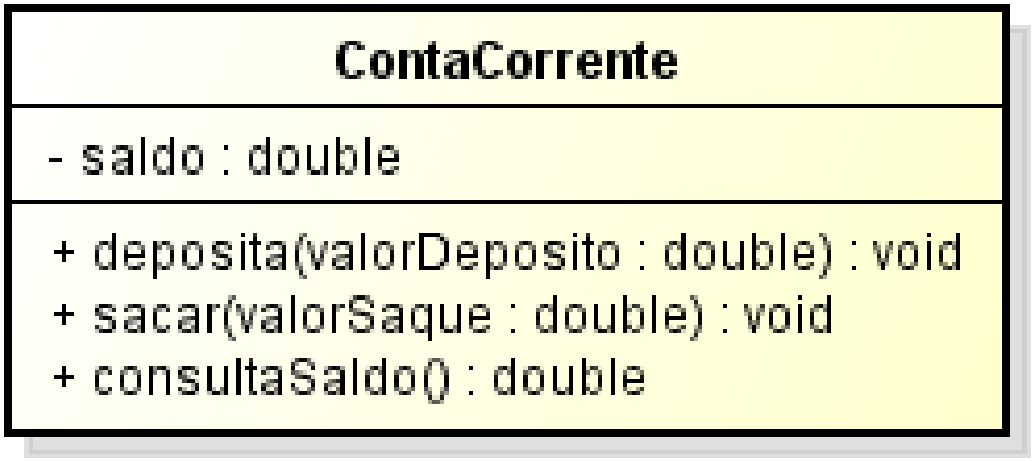
- **Unidade**
  - Unidade é a menor parte testável de uma aplicação. Em uma linguagem de programação orientada a objetos como o Java, a menor unidade é um método.

Testes de Unidade

- Quando criamos um teste de unidade, simulamos a execução de métodos da classe a ser testada.
- Fazemos isso passando parâmetros (no caso de ser necessário) ao método testado e definimos o resultado que esperamos.
- Se o resultado for igual ao que definimos como esperado, o teste passa. Caso contrário, falha.

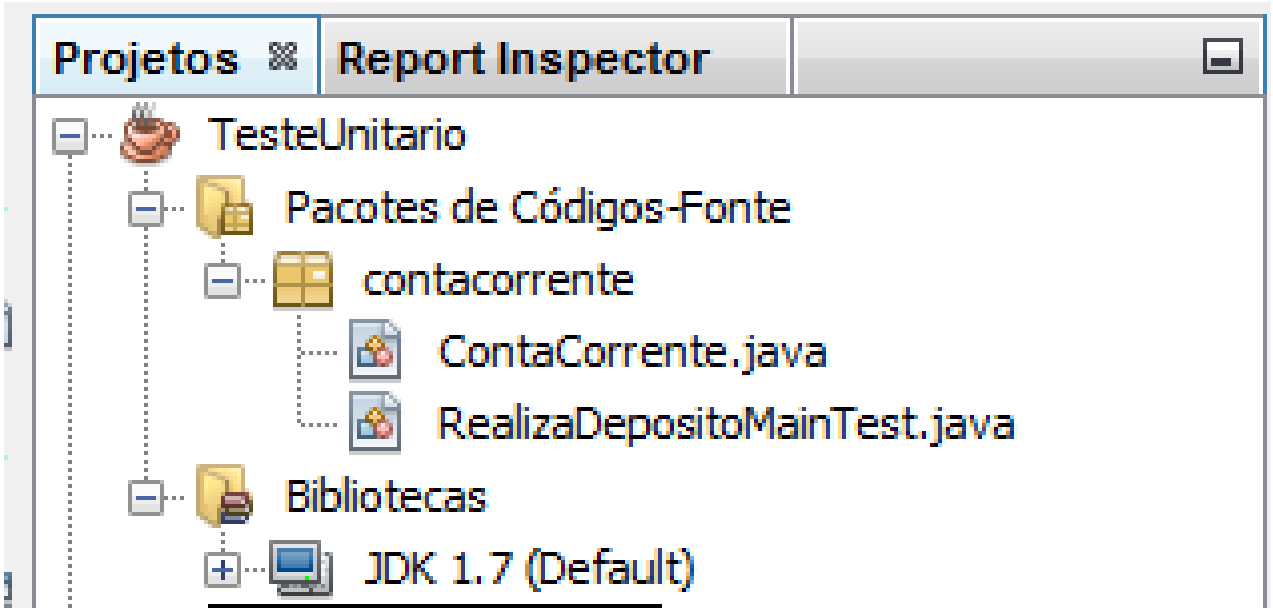
Primeiros Testes

- Realizar depósitos de qualquer valor
- Realizar saque apenas quando o saldo for suficiente.



Primeiros Testes

- Criar um Novo Projeto Java
- Criar a Classe ContaCorrente



Primeiros Testes

```
11  public class ContaCorrente {
12
13      private double saldo;
14
15      public void depositar(double valorDeposito) {
16          saldo += valorDeposito;
17          System.out.println("Deposito realizado.");
18      }
19
20      public void sacar(double valorSaque) {
21          if (valorSaque <= this.saldo) {
22              saldo -= valorSaque;
23              System.out.println("Saque Realizado.");
24          } else {
25
26              System.out.println("Saldo insuficiente. Saque não realizado");
27          }
28      }
29
30      public double consultaSaldo() {
31          return saldo;
32      }
33  }
```

Primeiros Testes

- Inicialmente, testamos o método depositar com um método main.

```
12  public class RealizaDepositoMainTest {
13      public static void main(String[] args) {
14          ContaCorrente cc = new ContaCorrente();
15          cc.depositar(100.0);
16          double saldoAtual = cc.consultaSaldo();
17
18          System.out.println("Saldo Atual: "+ saldoAtual);
19
20      }
21  }
```

```
Saída
Console do Depurador | TesteUnitario (run)
run:
Deposito realizado.
Saldo Atual: 100.0
CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo)
```

JUnit

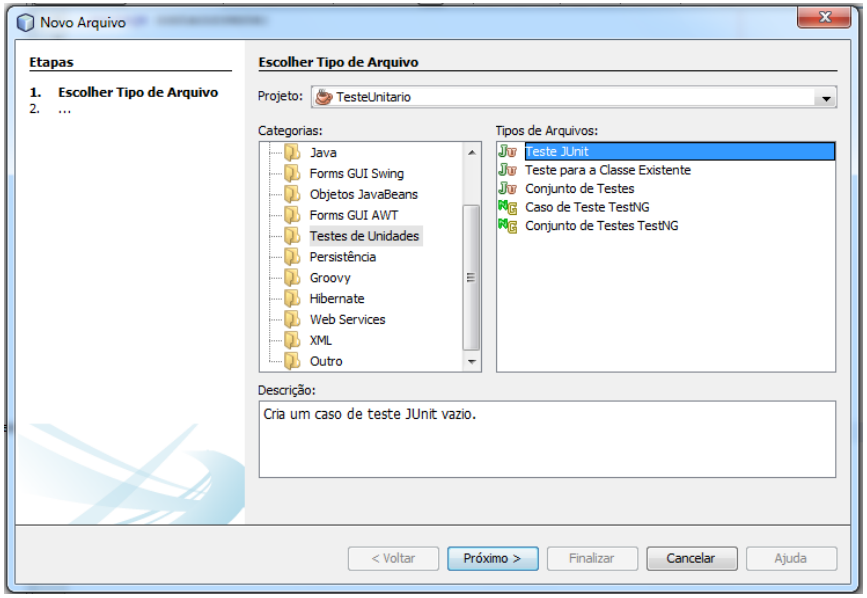
- O JUnit (junit.org) é um framework muito simples para facilitar a criação destes testes de unidade e em especial sua execução.
- Ele possui alguns métodos que tornam seu código de teste bem legível e fácil de fazer as asserções.

JUnit

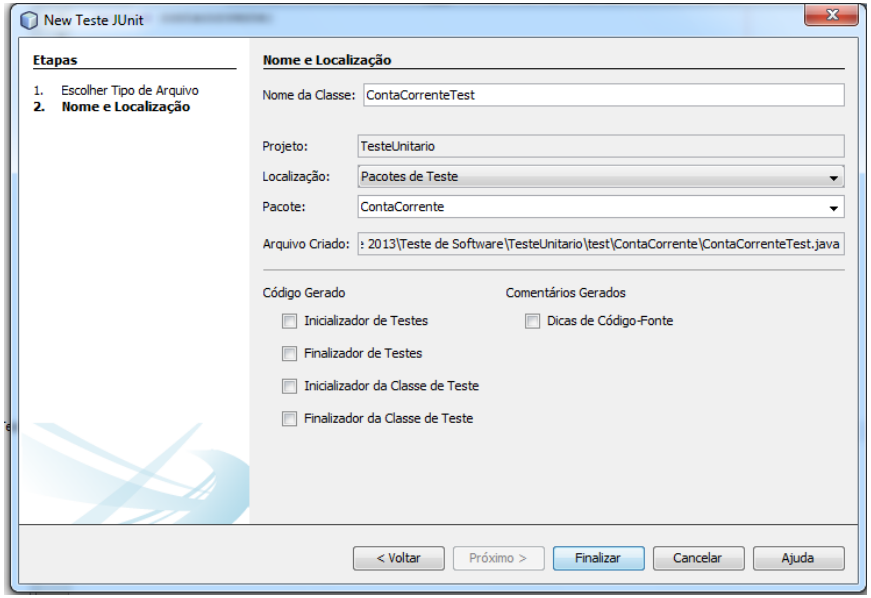
- **Asserção**:
  - é uma afirmação: alguma condição que em determinado ponto de execução você quer garantir que é verdadeira.
  - Se esta não for verdade, o teste deve indicar uma falha, a ser reportada para o programador, indicando um possível bug.

Executando testes com Junit e NetBeans

- Clicando com o botão direito do mouse sobre pacotes de código fonte > Novo > outros



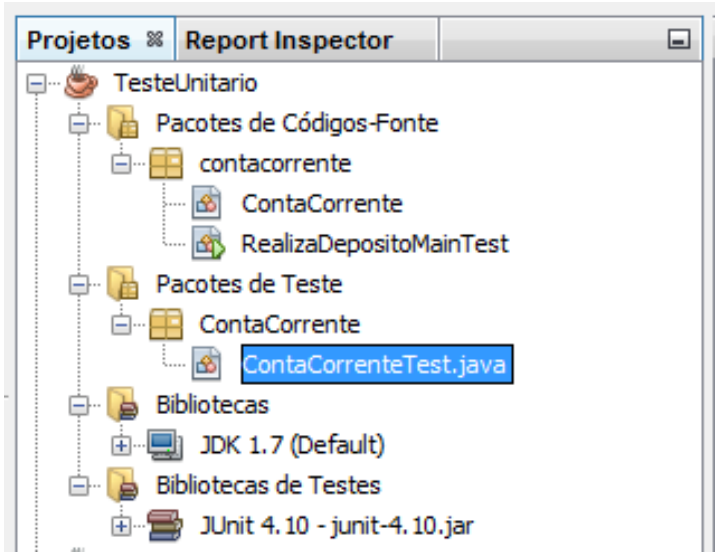
Passo 1: Selecione Teste JUnit



Passo 2:
Nome da Classe: ContaCorrenteTest
Localização: Pacotes de Teste
Pacote: ContaCorrente

Executando testes com Junit e NetBeans

- No projeto, foi criado um novo diretório de pacotes, Pacotes de Teste, com o pacote ContaCorrente que contem a classe ContaCorrenteTest.
- Esta Classe, faz uso do Framework de Teste Unitário Junit.




```
5   package ContaCorrente;
6
7   import org.junit.Test;
8   import static org.junit.Assert.*;
9
10
11
12
13
14  public class ContaCorrenteTest {
15
16      public ContaCorrenteTest() {
17      }
18  }
19
```

Executando testes com Junit e NetBeans

- Escrevendo o primeiro teste unitário com JUnit

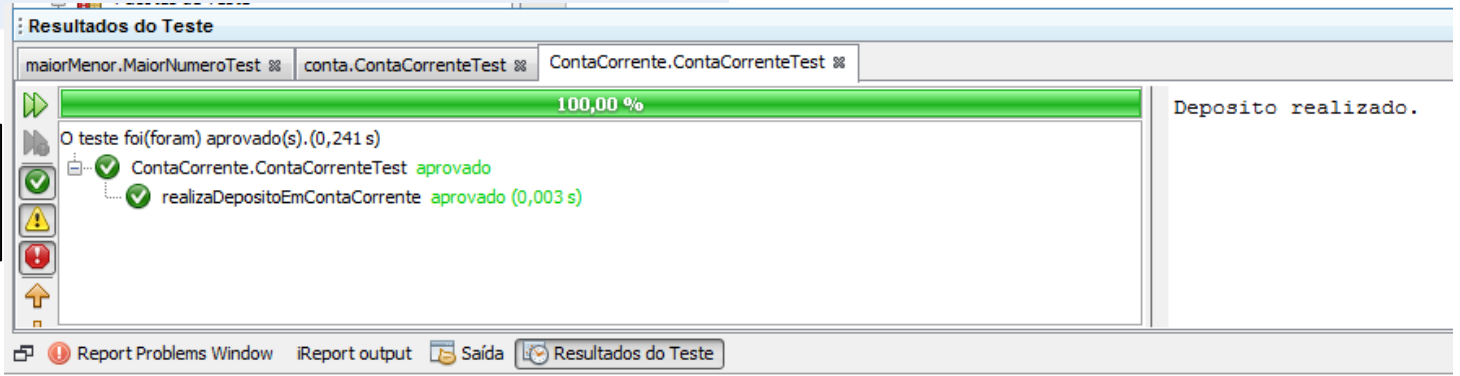
```
7  import contacorrente.ContaCorrente;
8  import org.junit.Test;
9  import static org.junit.Assert.*;
10
11 public class ContaCorrenteTest {
12
13     @Test
14     public void realizaDepositoEmContaCorrente(){
15         double valorDeposito = 200.0;
16         double saldoEsperado = 200.0;
17         ContaCorrente cc = new ContaCorrente();
18         cc.depositar(valorDeposito);
19         double saldoDaConta =  cc.consultaSaldo();
20
21         assertEquals(saldoEsperado, saldoDaConta, 0.0001);
22     }
23 }
24
```

Anotação Junit para definir um método de teste

Define um cenário de teste

Método estático da classe Assert para validar o teste

Resultado do Teste



Executando testes com Junit e NetBeans

- **Convenção e Anotação**
  - Para cada classe, teremos uma classe correspondente, por convenção, com o sufixo Test que contará todos os testes relativos aos métodos dessa classe.
  - Em vez de um main, criamos um método com nome expressivo para descrever a situação que ele está testando.
  - Anotamos este método com @Test, que fará com que o JUnit saiba no momento de execução que aquele método deve ser executado.

Executando testes com Junit e NetBeans

- **Asserções**
  - Uma asserção é uma verificação. Ela é realizada através dos métodos estáticos da classe Assert, importada do org.junit.

```
9  import static org.junit.Assert.*;
```

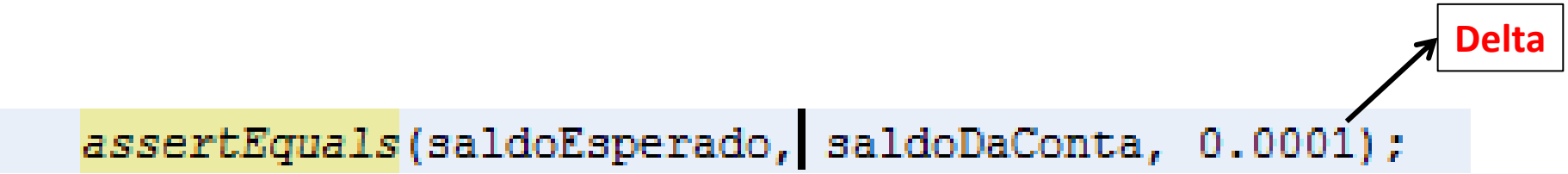
```
assertEquals(saldoEsperado, saldoDaConta, 0.0001);
```

  - O primeiro argumento é o que chamamos de expected, e ele representa o valor que esperamos para argumento seguinte (chamado de actual). Se o valor real for diferente do esperado, o teste não passará e uma barrinha vermelha será mostrada, juntamente com uma mensagem que diz:

expected <valor esperado> but was <o que realmente deu>

Executando testes com Junit e NetBeans

- **Double é inexato**
  - Double é um tipo de dado inexato ao trabalharmos com arredondamentos. Porém, diversas vezes, precisamos comparar o double esperado e o valor real, sem nos preocupamos com diferenças de arredondamento quando elas são **muito** pequenas.
  - O JUnit trata esse caso adicionando um terceiro argumento, que só é necessário quando comparamos valores double ou float. Ele é um delta que se aceita para o erro de comparação entre o valor esperado e o real.



Passos para criação de testes unitários

- De maneira generalizada o desenvolvedor
  1. Pensar primeiro em um cenário (um valor a ser testados)
  2. Executa a ação (executa os método da classe)
  3. Valida a saída (define a asserção)

Passos para criação de testes unitários


```
7  import contacorrente.ContaCorrente;
8  import org.junit.Test;
9  import static org.junit.Assert.*;
10
11 public class ContaCorrenteTest {
12
13     @Test
14     public void realizaDepositoEmContaCorrente(){
15         double valorDeposito = 200.0;
16         double saldoEsperado = 200.0;
17         ContaCorrente cc = new ContaCorrente();
18         cc.depositar(valorDeposito);
19         double saldoDaConta =  cc.consultaSaldo();
20
21         assertEquals(saldoEsperado, saldoDaConta, 0.0001);
22     }
23 }
24
```

Define um cenário de teste

Executa a ação

Valida a saida