

FIAP

FIAP

SLIDER



COMPLIANCE & QUALITY ASSURANCE

Prof. M.Sc. Felipe Desiglo Ferrare
proffelipe.ferrare@fiap.com.br

Introdução aos Testes Unitários

Aula 4

Cronograma

- Maven e JUnit
- Características dos Testes Unitários
- Criando Primeiros Testes Unitários
- Mockito

Maven

Dependências

- **Maven**: Ferramenta de gerenciamento de Dependências

<https://maven.apache.org/>

Linha de comando ou pela IDE
Arquivo de configuração (pom.xml)

Ex.:

`mvn clean install`

`mvn compile`

`mvn test`



JUnit

Dependências

- **JUnit**: Library de desenvolvimento de testes Unitários (Java)

<https://junit.org>

- Última versão 5 (Não compatível com 4 e 3
Diferenças de sintaxe)



JUnit

Dependencias

- Temos que considerar qual ferramenta estamos usando
- Diferenças entre versões e Linguagens



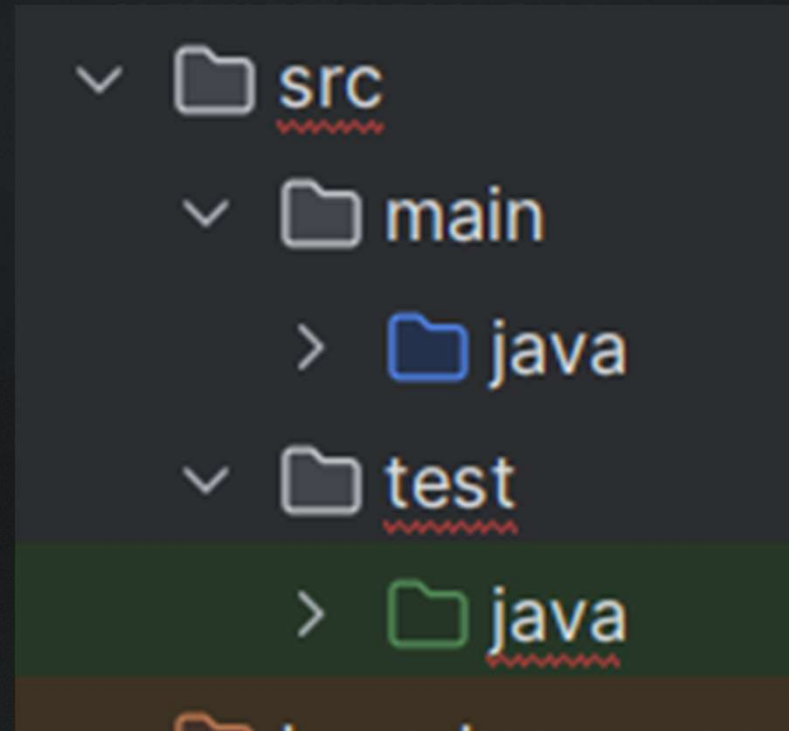
- Arquivo pom.xml

```
<dependency>  
  <groupId>org.junit.jupiter</groupId>  
  <artifactId>junit-jupiter-api</artifactId>  
  <version>5.10.2</version>  
  <scope>test</scope>  
</dependency>
```


JUnit

Maven

- Arquivos de Teste
- Testes em um Diretório separado



Testes Unitários


Boas Práticas

- Organizar os testes
- Manter cada teste simples
- Para cada Ação Tentar validar o que foi feito

Testes Unitários

Início

- Tag **@Test**
- Marca que o método é um Teste



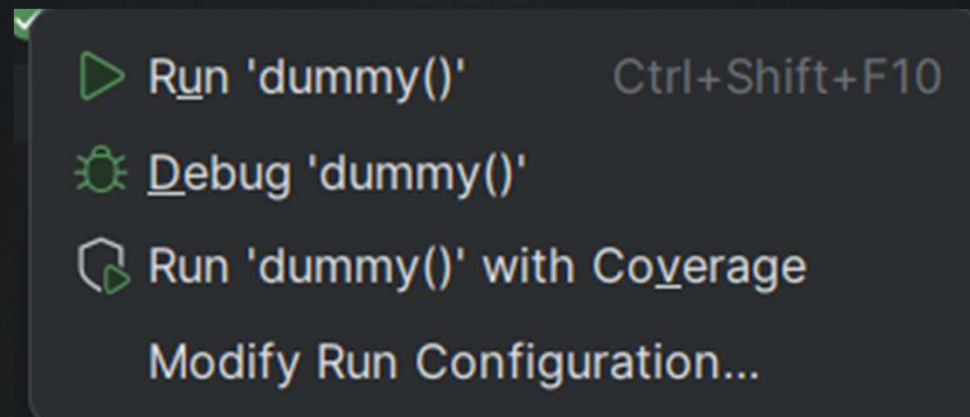
```
@Test  
public void dummy() {  
    |  
}  

```

Testes Unitários

Início


- Possível Executar testes unitários de maneira Individual ou Grupo
- Debug de Testes, Validar Cobertura




Testes Unitários

Cobertura

- **Análise do que está sendo testado**

Coverage CalcTest.dummy x			
			
Element ^	Class, %	Method, %	Line, %
▼ com.fiap	100% (1/1)	50% (1/2)	33% (2/6)
Ⓢ Calc	100% (1/1)	50% (1/2)	33% (2/6)

Coverage CalcTest x			
			
Element ^	Class, %	Method, %	Line, %
▼ com.fiap	100% (1/1)	100% (2/2)	100% (6/6)
Ⓢ Calc	100% (1/1)	100% (2/2)	100% (6/6)

Testes Unitários

Tags

- Com a marcação `@Tag` é possível dar uma tag em testes cases para executá-los em conjunto

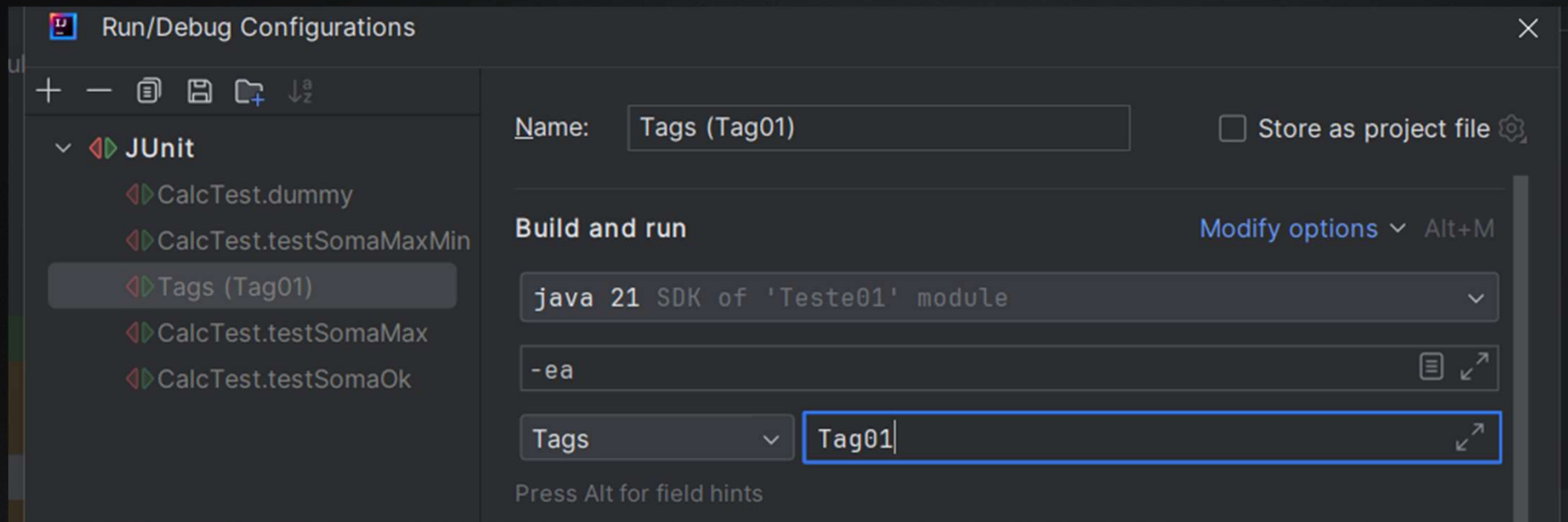
```
@Test
@Tag("TagNameDummy")
public void dummy() {

}
```

Testes Unitários

Tags

- É possível colocar para executar somente os testes com uma tag (IDE, linha de comando)



Testes Unitários

Outras Opções

- Comandos possíveis

Comando	Descrição
@DisplayName	Altera o nome que o test vem nos resultados
@Order	Define um valor de ordem nos testes (má-pratica)
@Disabled	Desativa o Test na execução geral
@Timeout	Seta um tempo limite para execução

Testes Unitários

Outras Opções

- Comandos possíveis

@BeforeEach	Antes de cada Test
@AfterEach	depois
@BeforeAll	Antes dos Testes
@AfterAll	Depois dos Testes

Testes Unitários

Outras Opções

- Comandos possíveis

@BeforeEach	Antes de cada Test
@AfterEach	depois
@BeforeAll	Antes dos Testes
@AfterAll	Depois dos Testes

Testes Unitários

Outras Opções

- Validando

<code>assertArrayEquals</code>	Verifica se as matrizes passadas nos parâmetros <i>expected</i> e <i>actual</i> são iguais.
<code>assertEquals</code> <code>assertNotEquals</code>	Verifica se os objetos passados nos parâmetros <i>expected</i> e <i>actual</i> são iguais ou diferentes.
<code>assertTrue</code> <code>assertFalse</code> <code>e</code>	Verifica se dada condição retorna o booleanos Verdadeiro (<i>True</i>) ou Falso (<i>False</i>).
<code>assertNull</code> <code>assertNotNull</code>	Verifica se um dado objeto é ou não nulo (<i>null</i>).
<code>assertThrows</code> <code>assertDoesNotThrow</code>	Permite verificar se um executável (<i>executable</i>) lança uma exceção do tipo especificado, ou não o lança nenhuma exceção.
<code>assertAll</code>	Permite a criação de asserções agrupadas, onde todas são executadas e suas falhas reportadas em conjunto.
<code>fail</code>	Ao ser executado, atribui falha ao teste imediatamente, adicionando mensagem opcional de falha.

Exercício

Calculadora

- Implementar a Interface (Calculadora) e seus testes

```
package com.fiap;  
  
public interface Calculator {  
    public int soma(int valor1, int valor2) throws Exception;  
    public int divisaoInt(int valor1, int valor2) throws Exception;  
    public float divisao(int valor1, int valor2) throws Exception;  
    public int multiplicacao(int valor1, int valor2) throws Exception;  
    public int subtracao(int valor1, int valor2) throws Exception;  
}
```


Mocks

Isolando Código

- Idealmente para garantir que nossos testes unitários estão colocado somente o código do método ou do componente que estamos testando, podemos isolar esse código com o uso de Mocks, que são chamadas para objetos que criamos somente para aquela execução de testes, onde podemos definir os comportamentos e validar somente a lógica de forma isolada.

Mocks

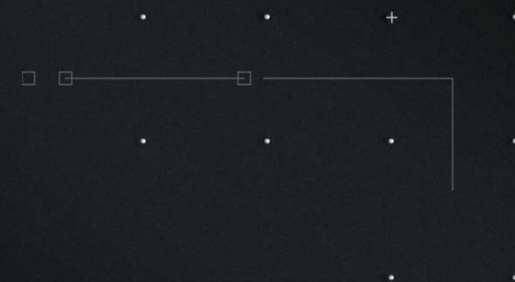
Ferramenta Mockito

- <https://site.mockito.org/>

Tasty mocking framework for unit tests in Java



Obrigado!



FIAP