




DIGITAL
INNOVATION
ONE

Testes com Java

Junit 5

 Rodrigo Tassini

Software Engineer - CMA



DIGITAL
INNOVATION
ONE

Objetivos da Aula

1. Overview

2. Arquitetura

3. Annotations

4. Asserts e Assumption



DIGITAL
INNOVATION
ONE

JUnit 5

Testes com Java



JUnit 5

```
1 <dependency>
2   <groupId>org.junit.jupiter</groupId>
3   <artifactId>junit-jupiter-engine</artifactId>
4   <version>5.1.0</version>
5   <scope>test</scope>
6 </dependency>
```



JUnit 5

Annotation	Description
@Test	Denotes that a method is a test method. Unlike JUnit 4's @Test annotation, this annotation does not declare any attributes, since test extensions in JUnit Jupiter operate based on their own dedicated annotations. Such methods are <i>inherited</i> unless they are <i>overridden</i> .
@ParameterizedTest	Denotes that a method is a parameterized test . Such methods are <i>inherited</i> unless they are <i>overridden</i> .
@RepeatedTest	Denotes that a method is a test template for a repeated test . Such methods are <i>inherited</i> unless they are <i>overridden</i> .
@TestFactory	Denotes that a method is a test factory for dynamic tests . Such methods are <i>inherited</i> unless they are <i>overridden</i> .
@TestTemplate	Denotes that a method is a template for test cases designed to be invoked multiple times depending on the number of invocation contexts returned by the registered providers . Such methods are <i>inherited</i> unless they are <i>overridden</i> .
@TestMethodOrder	Used to configure the test method execution order for the annotated test class; similar to JUnit 4's @FixMethodOrder. Such annotations are <i>inherited</i> .



Junit 5

```
1  @Test
2  void lambdaExpressions() {
3      assertTrue(Stream.of(1, 2, 3)
4          .stream()
5          .mapToInt(i -> i)
6          .sum() > 5, () -> "Sum should be greater than 5");
7  }
```

```
1  @Test
2  void groupAssertions() {
3      int[] numbers = {0, 1, 2, 3, 4};
4      assertAll("numbers",
5          () -> assertEquals(numbers[0], 1),
6          () -> assertEquals(numbers[3], 3),
7          () -> assertEquals(numbers[4], 1)
8      );
9  }
```



Junit 5

```
1  @Test
2  void trueAssumption() {
3      assumeTrue(5 > 1);
4      assertEquals(5 + 2, 7);
5  }
6
7  @Test
8  void falseAssumption() {
9      assumeFalse(5 < 1);
10     assertEquals(5 + 2, 7);
11 }
12
13 @Test
14 void assumptionThat() {
15     String someString = "Just a string";
16     assumingThat(
17         someString.equals("Just a string"),
18         () -> assertEquals(2 + 2, 4)
19     );
20 }
```