DIGITAL
INNOVATION
ONE

Objetivos da Aula

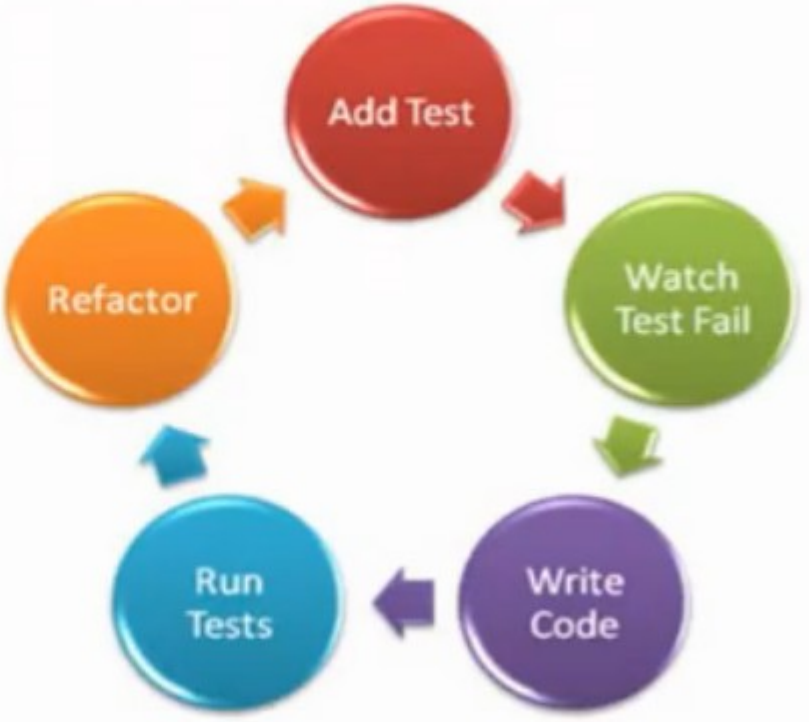
1. O que é TDD?

2. Benefícios

3. Primeiros passos

DIGITAL INNOVATION ONE

TDD

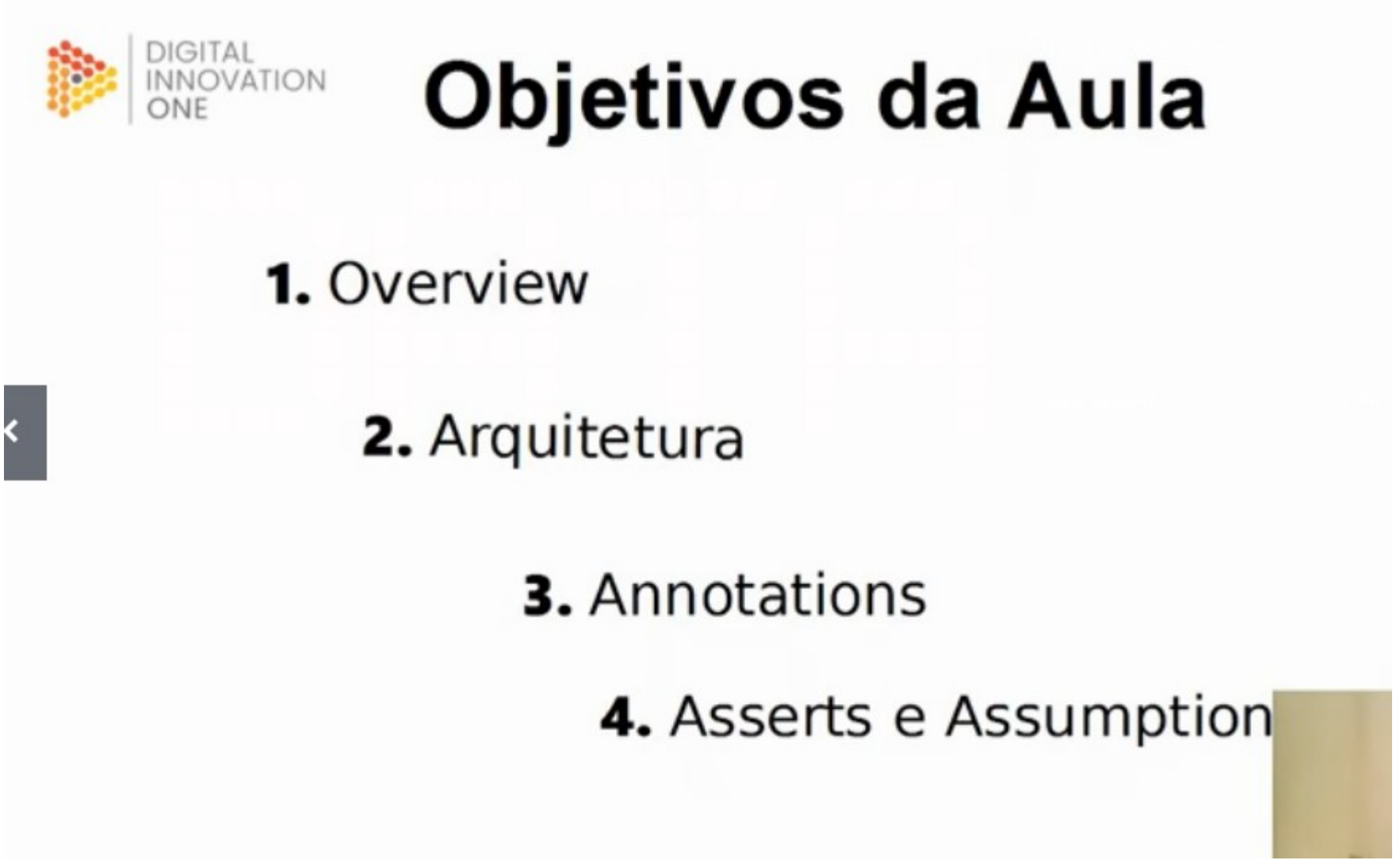


DIGITAL
INNOVATION
ONE

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

```xml
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>5.1.0</version>
    <scope>test</scope>
</dependency>
```

Novidades do JUnit 5

@ParameterizedTests

Este é um novo tipo de teste utilizado com esta anotação que, como o nome já diz, é parametrizável. Você já se viu em uma situação em que está testando um mesmo método, mas este possui várias regras de negócios que pode ramificar em diferentes resultados dependendo dos parâmetros passados?

O JUnit 4 não tinha essa funcionalidade por padrão, sendo necessário utilizar outras bibliotecas. Agora é totalmente possível e fácil de realizar.

Outras novidades

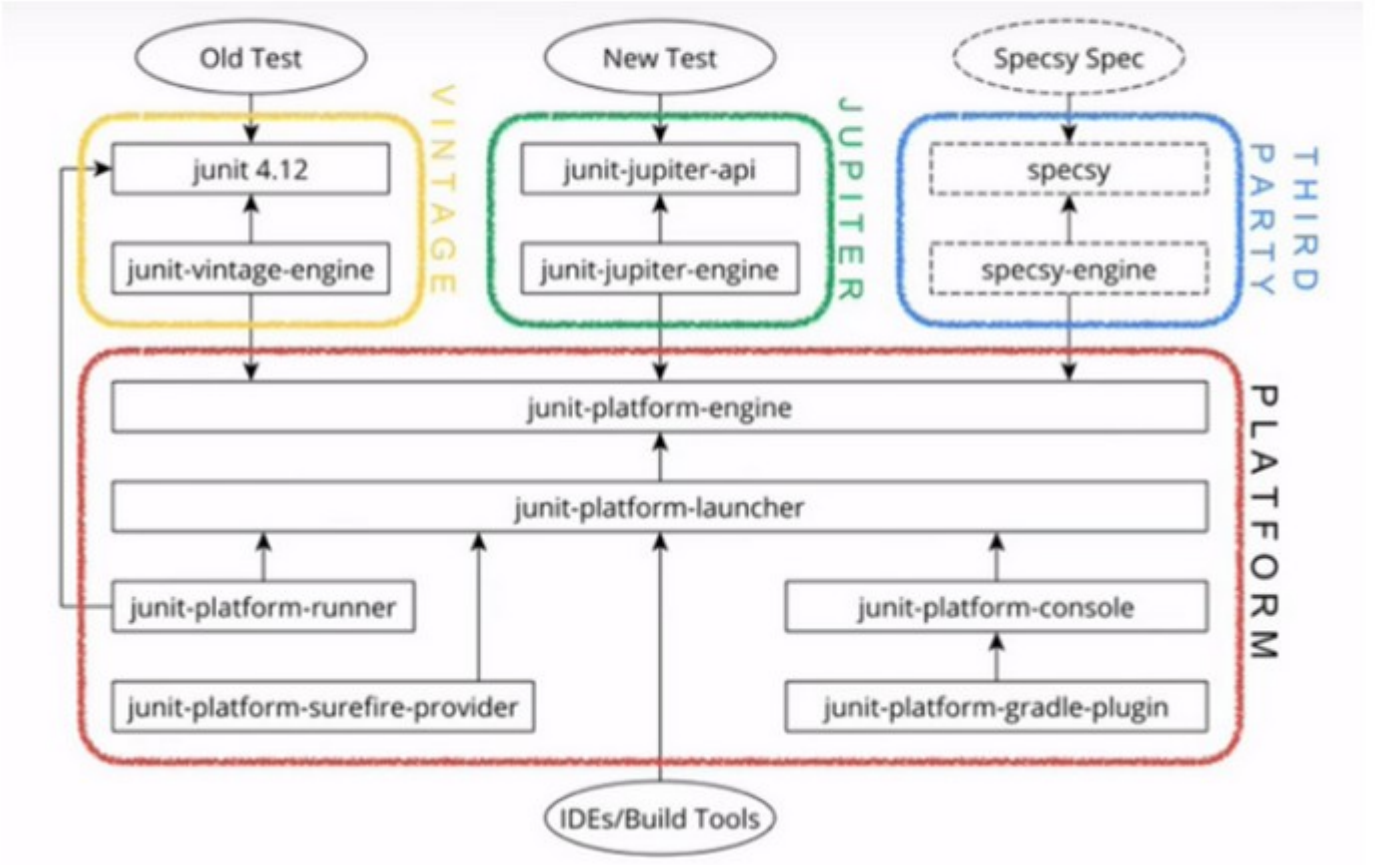
Há também outras novidades menores e outras grandes demais para explicar neste artigo, começando pelas menores:

- A anotação `@DisplayName` permite adicionar uma descrição melhor do que o nome dos métodos como no JUnit 4. Chega de ter nomes de métodos enormes para tentar descrever o teste e ainda assim não ficar muito bom para humanos entenderem.

```
@Test
@DisplayName("Deve calcular que 2 + 2 = 4")
public void calculateAddTest() {
 Integer result = 2 + 2;
 assertEquals(4, result);
}
```

- Algumas novas *assertions* foram introduzidas, além de `assertThrows()` e sua contrária `assertDoesNotThrow()` também há *assertions* para comparar `Iterable` como `assertIterableEquals()`. `assertLinesMatch()` compara duas listas de String, aceitando expressões regulares.
- Agora há como executar testes de forma condicional de acordo com uma série de contextos como OS (`@EnabledOnOs` & `@DisabledOnOs`), versão do Java (`@EnabledOnJre` & `@DisabledOnJre`), se uma propriedade de sistema está presente (`@EnabledIfSystemProperty`) ou utilizar sua própria lógica (`@EnabledIf`)

Uma visão macro da arquitetura com seus módulos pode ser encontrada na imagem abaixo:



DIGITAL INNOVATION ONE

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 *inherited* unless they are *overridden*.
`@ParameterizedTest`	Denotes that a method is a parameterized test. Such methods are *inherited* unless they are *overridden*.
`@RepeatedTest`	Denotes that a method is a test template for a repeated test. Such methods are *inherited* unless they are *overridden*.
`@TestFactory`	Denotes that a method is a test factory for dynamic tests. Such methods are *inherited* unless they are *overridden*.
`@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 *inherited* unless they are *overridden*.
`@TestMethodOrder`	Used to configure the test method execution order for the annotated test class; similar to JUnit `@FixMethodOrder`. Such annotations are *inherited*.

DIGITAL
INNOVATION
ONE

Junit 5

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

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

DIGITAL
INNOVATION
ONE

Junit 5

```java
@Test
void trueAssumption() {
    assumeTrue(5 > 1);
    assertEquals(5 + 2, 7);
}

@Test
void falseAssumption() {
    assumeFalse(5 < 1);
    assertEquals(5 + 2, 7);
}

@Test
void assumptionThat() {
    String someString = "Just a string";
    assumingThat(
        someString.equals("Just a string"),
        () -> assertEquals(2 + 2, 4)
    );
}
```

TDD e Testes Unitários com JUnit 5

Para que são usadas as novas Assumptions de JUnit 5?

- Para executar testes em uma ordem predefinida.
- Para executar testes quando o usuário determinar.
- Para executar testes apenas se determinadas condições forem atendidas. ✔
- Nenhuma das alternativas.
- Para executar testes apenas se determinadas condições não forem atendidas.

PRÓXIMA PERGUNTA

TDD e Testes Unitários com JUnit 5

Os módulos que compões o JUnit 5 são:

- JUnit Platform, JUnit Jupiter e JUnit Vintage ✔
- JUnit Platform, JUnit Mars e JUnit Vintage
- JUnit Platform, JUnit Jupiter e JUnit Modern
- JUnit 5, JUnit Jupiter e JUnit Vintage
- Nenhuma das alternativas.

PRÓXIMA PERGUNTA

TDD e Testes Unitários com JUnit 5

A versão 5 do JUnit tem o objetivo de oferecer suporte a qual versão do Java?

- 11
- 10
- 7
- 8 ✔
- 9

PRÓXIMA PERGUNTA

Por que TDD proporciona um código mais limpo?

- Por favorecer a criação de códigos estáveis.
- Por favorecer a criação de códigos com Design Patterns.
- Nenhuma das alternativas.
- Por favorecer a criação de códigos menores e testáveis. ✔
- Por favorecer a criação de códigos com programação orientada a objetos. ✖

PRÓXIMA PERGUNTA

TDD e Testes Unitários com JUnit 5

Como são chamados os pequenos passos do desenvolvimento no TDD?

- Step by step ✖
- Shortsteps
- First step
- Nenhuma das alternativas.
- Babysteps ✔

PRÓXIMA PERGUNTA

TDD e Testes Unitários com JUnit 5

TDD pode ser dividido em cinco passos, que são respectivamente:

- Nenhuma das alternativas.
- Teste, falhe, codifique, passe no teste, refatore. ✔
- Teste, codifique, falhe, refatore, passe no teste.
- Codifique, falhe, refatore, teste, passe no teste.
- Codifique, teste, falhe, refatore, passe no teste.

PRÓXIMA PERGUNTA

TDD e Testes Unitários com JUnit 5

A annotation @AfterEach de JUnit Jupiter, permite:

- Nenhuma das alternativas.
- Ser executado após cada classe de teste.
- Ser executado antes cada classe de teste.
- Ser executado após cada método de teste. ✔
- Ser executado antes cada método de teste.

PRÓXIMA PERGUNTA

TDD e Testes Unitários com JUnit 5

O JUnit platform é responsável pelo:

- Nenhuma das alternativas.
- Refatoração de estruturas de teste na JVM.
- Compilação de estruturas de teste na JVM.
- Codificação de estruturas de teste na JVM. ✖
- Lançamento de estruturas de teste na JVM. ✔

PRÓXIMA PERGUNTA

TDD e Testes Unitários com JUnit 5

Segundo Kent Beck, escrever teste antes, permite:

- Nenhuma das alternativas.
- Obter código com muito menos defeitos, porém com maior tempo de desenvolvimento.
- Obter código com muito menos defeitos, porém sem ganho de design.
- Obter código com muito menos defeitos e com um design muito mais limpo. ✔
- Obter código com poucos defeitos e com pouco esforço de desenvolvimento.

PRÓXIMA PERGUNTA

TDD e Testes Unitários com JUnit 5

O que é TDD?

- Uma metodologia de testes.
- Nenhuma das alternativas.
- Um framework de testes. ✖
- Uma metodologia de desenvolvimento de software. ✔
- Um framework de orientação a objetos.

FINALIZAR