Unit 5

- Mundanças -

JUnit5

Necessita do Java 8

Suporte a expressões Lambda. (n) -> n*n

Principal mudança foi arquitetural.

Passou a ser composto por 3 projetos.

JUnit 5 = JUnit Platform + JUnit Jupiter + JUnit Vintage

JUnits - Partes

- **JUnit Platform** - suporte ao desenvolvimento de frameworks de teste.
- **JUnit Jupiter** - permite construção de tests no JUnit 5.
- **JUnit Vintage** - suporta a execução de testes JUnit 3 e JUnit 4 (backward compatibility).

JUnit5 x JUnit4
Algumas Diferenças

JUNIT 4	JUNIT 5
`@Test`	`@Test`
`@BeforeClass`	`@BeforeAll`
`@AfterClass`	`@AfterAll`
`@Before`	`@BeforeEach`
`@After`	`@AfterEach`
`@Ignore`	`@Disabled`

Testes Parametrizados no JUnit5

https://junit.org/junit5/docs/current/user-guide/#writing-tests-parameterized-tests

```
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.ValueSource;

import static
  org.junit.jupiter.api.Assertions.assertNotNull;

@DisplayName(“Testes Param Exemplo Classe")
class ExemploTest {

    @DisplayName(“Nome do Teste")
    @ParameterizedTest
    @ValueSource(strings = {"Hello", "World"})
    void metodoTesteParam(String message) {
        assertNotNull(message);
    }

}

//Apenas 1 parâmetro
```

```
@ValueSource pode ser:
                        short
                        byte
                        int
                        long
                        float
                        double
                        char
                        java.lang.String
                        java.lang.Class
```

```java
@ParameterizedTest
@MethodSource("stringIntAndListProvider")
void testWithMultiArgMethodSource
            (String str, int num, List<String> list) {

        assertEquals(3, str.length());
        assertTrue(num >=1 && num <=2);
        assertEquals(2, list.size());

}

static Stream<Arguments> stringIntAndListProvider() {

    return Stream.of(
        arguments("foo", 1, Arrays.asList("a", "b")),
        arguments("bar", 2, Arrays.asList("x", "y"))
    );

}

 //Método gerador
```

```
@ParameterizedTest
@CsvFileSource(resources = "/arq.csv", numLinesToSkip = 1)
void test(String first, int second) {
    assertNotNull(first);
    assertNotEquals(0, second);
}
```

- - - - - - - - - - - -

arq.csv

```
Country, reference
Sweden, 1
Poland, 2
"United States of America", 3
```

//Arquivo com dados de teste

No curso focaremos no

JUnit4

pois algumas APIs e ferramentas que utilizaremos ainda não suportam o JUnit5

Extra:
Test Driven Development (TDD)

Profa. Roberta Coelho
Departamento de Informática e Matemática Aplicada - DIMAp

Objetivos

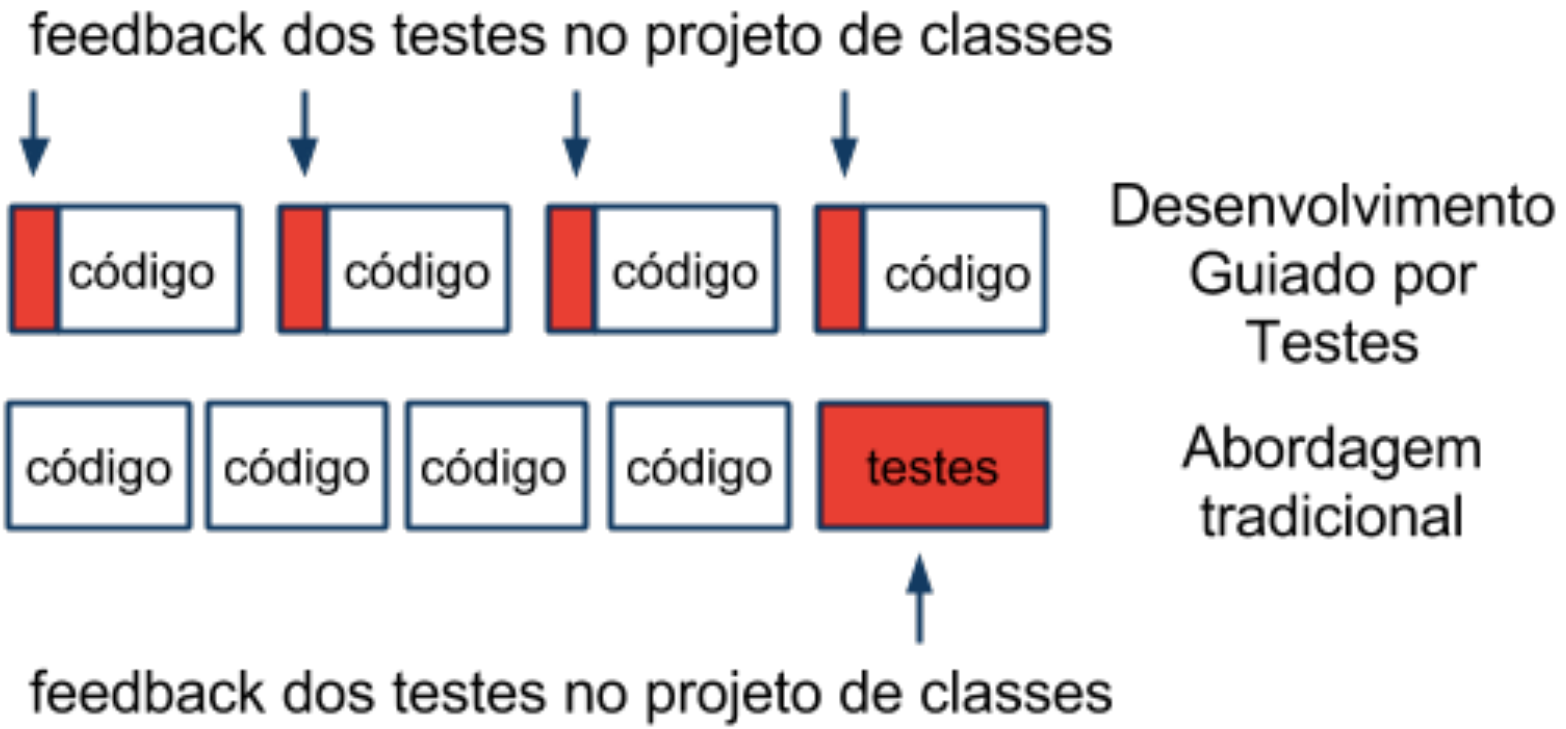
- Discutir sobre TDD na Pratica!

Benefícios Prometidos

1 - Feedback

a granulosidade fina do teste-então-codifique permite contínuo feedback ao desenvolvedor.

1 - Feedback


feedback dos testes no projeto de classes
código
código
código
código
Desenvolvimento Guiado por Testes
código
código
código
código
testes
Abordagem tradicional
feedback dos testes no projeto de classes

Benefícios Prometidos

2 - Ajuda no design da solução

Os testes servem de especificação do comportamento dos métodos (e possíveis exceções).

Benefícios Prometidos

3 - Inclusão x Detecção

Reduzir o tempo entre a inclusão e a detecção do bug.

Benefícios Prometidos

4 - Testes Automatizados

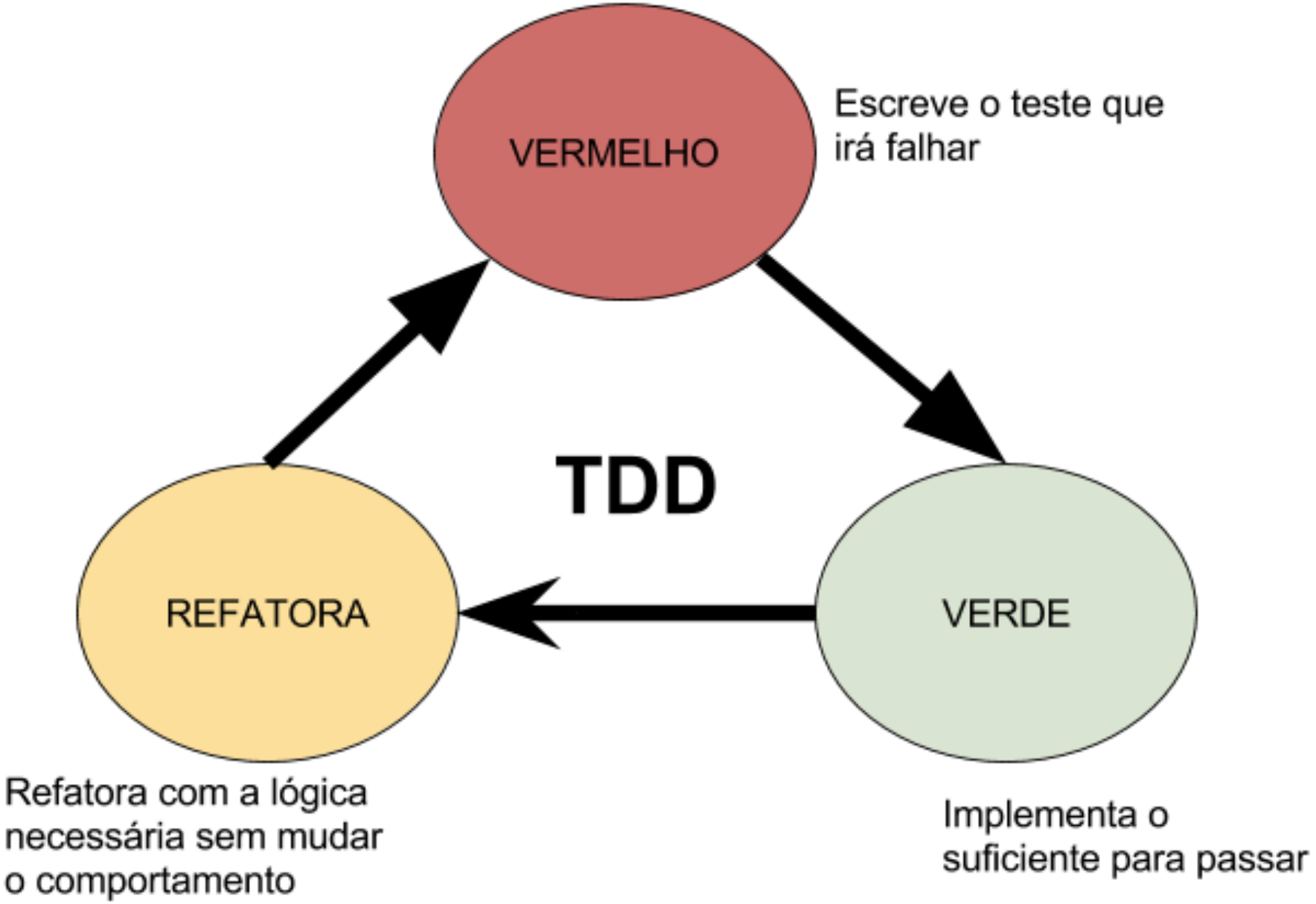
Será criado um suite testes unitários para realizar testes de regressão.

Algumas métricas da indústria

Table 3: Project A- Outcome Measures

Metric Description	Value
Actual defects/KLOC (using TDD)	X
Defects/KLOC of comparable team in org but not using TDD	2.6X
Increase in time taken to code the feature because of TDD (%) [Management estimates]	25-35%

Ciclo TDD


VERMELHO
Escreve o teste que irá falhar
TDD
REFATORA
VERDE
Refatora com a lógica necessária sem mudar o comportamento
Implementa o suficiente para passar

Quando usar TDD?

- Quando se precisa entender melhor o que deve ser implementado.

- funcionalidades críticas do sistema.


Mão na Massa

Mão na Massa 7

Vamos implementar a classe tdd.ContaCorrente a partir de uma suite de testes pre-existente.

Na prática no TDD o próprio desenvolvedor constrói os testes, esta suite foi definida para tornar a atividade mais rápida.

Resumo

- JUnit diferentes versões

- Mão na massa

- TDD