Teste de Software

Prof. Dr. Bruno Queiroz Pinto

TDD - *Test Driven Development*

- Estamos muito acostumados a implementar o código de produção e testá-lo ao final.

- Mas será que essa é a **única** maneira de desenvolver um projeto?

- O TDD ou "desenvolvimento orientado a teste" propõe começar pelos testes. Além disso, orienta que seja implementado sempre o código mais simples que resolva o problema.

TDD - *Test Driven Development*

- É um **método de desenvolver software**. Consiste em um desenvolvimento guiado pelos testes.

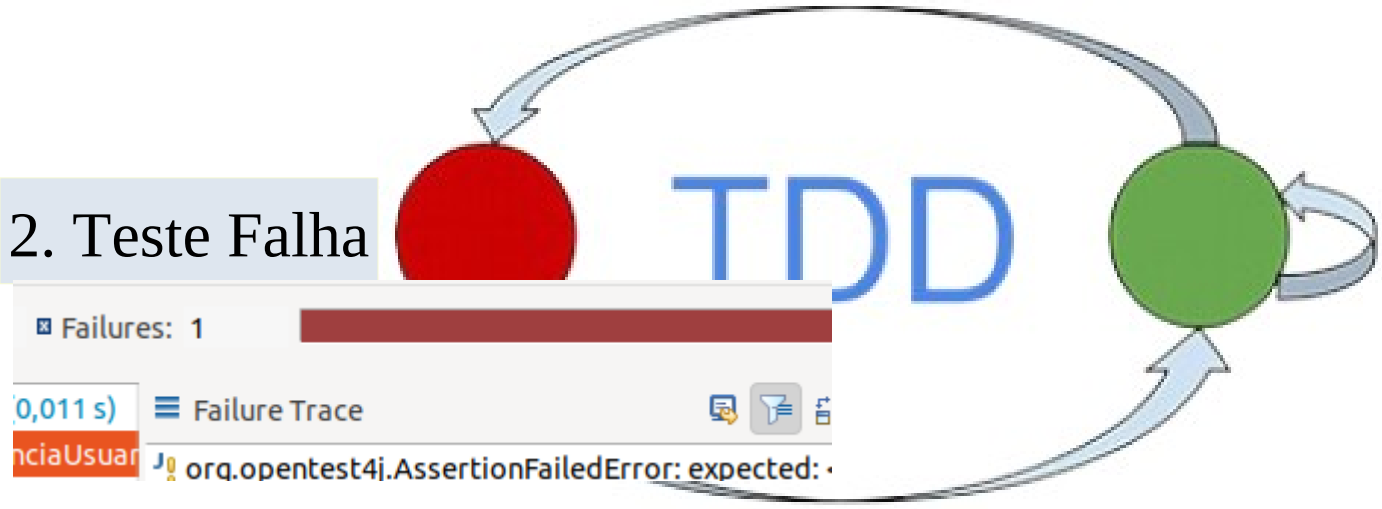
- Processo básico:
  - Escreva o teste como esperado (naturalmente que ele ainda estará falhando);
  - Implemente o código necessário para que o teste passe;
  - Refatore o código para melhorá-lo.

TDD - *Test Driven Development*

```
@Test
public void naoAceitLancesEmSequenciaUsuario() {
leilao.propoe(new Lance(joao, 1000));
leilao.propoe(new Lance(joao, 2000));

assertEquals(1, leilao.getLances().size());
assertEquals(1000.0, leilao.getLances().get(0).getValor(), 0.000001);
}
```

1. Escrevemos um teste



2. Teste Falha

3. Modificamos o código para ele passar no teste

```
public void propoe(Lance lance) {
if (lances.isEmpty() ||
!lances.get(lances.size()-1).getUsuario().equals(lance.getUsuario())) {
lances.add(lance);
}
}
```

4. Refatorar o código para melhorá-lo

TDD - *Test Driven Development*

- Vantagens:
  - Se sempre escrevermos o teste antes, garantimos que todo nosso código já "nasce" testado;
  - Temos segurança para refatorar nosso código, afinal sempre refatoramos com uma bateria de testes que garante que não quebraremos o comportamento já existente;

TDD - *Test Driven Development*

- Vantagens:
  - Como o teste é a primeira classe que usa o seu código, você naturalmente tende a escrever código mais fácil de ser usado e, por consequência, mais fácil de ser mantido.
  - Tende a melhorar o design do código, pois o código deverá ser testável.

TDD - *Test Driven Development*

- Como toda prática de ESOF, este método deve ser usado no momento certo. TDD faz sentido ao implementar novas funcionalidades, ao corrigir bugs, ao trabalhar em códigos complexos etc.

- Em códigos extremamente simples, talvez a prática de TDD não seja necessária. Mas lembre-se: cuidado para não achar que "tudo é simples", e nunca praticar TDD.