
academia
CHaORDiC

Test Driven Development

Danilo Queiroz | danilo@**chaordic**.com.br

Objetivo

Entender **porque**, **quando** e **como** usar **TDD**.

academia CHAORDIC

Por que, quando e como?

- Porque você **DEVE** sempre testar seu código
- **Sempre**
- Escrevendo os testes **primeiro**

*The tests you write **after are defense**. The tests you write **first are offense**.*

-- Uncle Bob

The Clean Coder

academia CHaORDiC

As 3 regras do TDD

- Você **não pode** escrever **nenhum código de produção** até você ter escrito primeiro **um teste que esteja falhando**
- Você **não pode** escrever mais código de teste do que **precise para o teste falhar** - e não compilar é falhar
- Você **não pode** escrever mais código de produção do que seja **suficiente para o seu teste atual passar**

academia CHAORDIC

Exercício #1

CHaORDiC you'll like

Seguindo as 3 regras do TDD, implementem um método que dado um número inteiro, retorna sua representação em números romanos

```
public static String toRoman(int number) {
  return "";
}
```

Quem seguiu **estritamente** as 3 regras? Quais foram **dificuldades**?

academia CHAORDIC

TDD e Boas Práticas

- Teste também é código
  - Todas as regras se aplicam!
- Se seus testes não fáceis de manter, seu código não será fácil de manter

academia CHAORDIC

TDD e Boas Práticas

- Testes devem ser focados - devem testar **uma** coisa
- Use nomes **descritivos** - e seja consistente
- Testes de unidade devem ser **rápidos**
- Evite testes com **zero delta coverage**

academia CHAORDIC

Talk is cheap. Show me the code!

-- Linus Torvalds

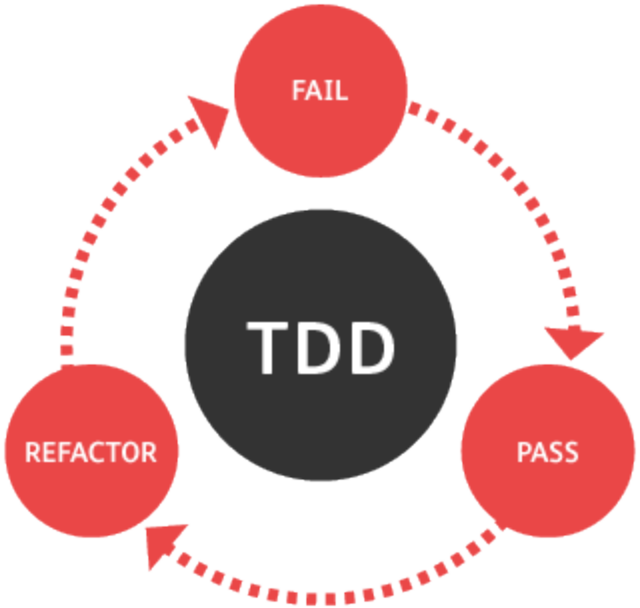
academia CHAORDIC

Por que escrever **testes**?

... E por que fazer isso **primeiro**?

academia CHAORDIC

Ciclo do TDD



academia CHAORDIC

O objetivo do **TDD** é o **DESIGN**, não os testes

TDD + Refactoring = Evolutionary Design

academia CHAORDIC

Exercício #2

CHaORDiC you'll like

Implementar uma classe que conta ocorrências (palavras, letras, ...) em um arquivo.

```
public class OccorrencesCounter {
  public Map<String, Integer> count() {
    return emptyMap();
  }
}
```

- Princípios de Design
- Estilos de testes
- Uso de Mocks
- Código Legado e TDD

academia CHAORDIC

Parte I

- Ciclo do TDD
- Boas práticas
- Algumas dificuldades
- TDD como técnica de design

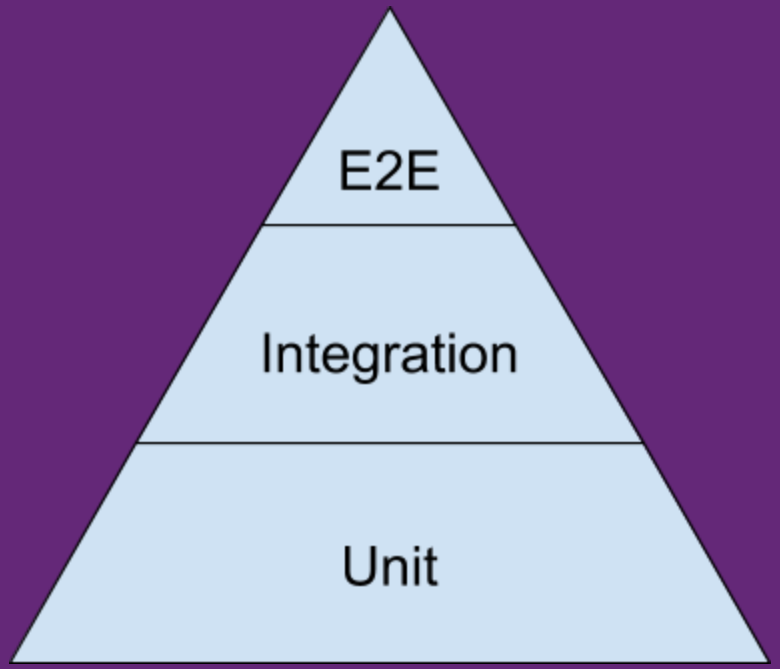
academia CHAORDIC

Princípios de design

- KISS
- YAGNI
- Single responsability (SOLID)
- DRY

academia CHAORDIC

Tipos de teste



academia CHAORDIC

Estilos de testes

- Verificação de estado
  - Foco é no resultado gerado
- Verificação de comportamento/interação
  - Foco nas interações do objeto com outros

Qual abordagem utilizar?

academia CHAORDIC

*Any decent answer to an interesting question begins, "**it depends...**"*

-- Kent Beck

academia CHaORDiC


Let’s see some **code**!
academia CHaORDiC

Utilizando Mocks

- Mocks são poderosos
    - Permitem desacoplar/isolar e dão flexibilidade
- Cuidado com uso excessivo
    - Mocks podem indicar um “code-smell”

academia CHAORDIC

*To be a **successful** mockist, you must **dislike mocks**.*

-- @Avid

Exercício #3

CHaORDiC you'll like

Implementar uma camada de cache para o seguinte DAO

```
public interface AccountDAO {
  public Account getAccountById(String id);

  public Account store(Account account);
}
```

Vamos falar de código **legado** e **TDD**...

academia CHAORDIC

*To me, **legacy code** is code simply code **without tests**.*

-- Michael Feathers

academia CHAORDIC

Código Legado e TDD

- Crie testes para o código que você prentende modificar
  - Ache a ramificação mais curta
  - Isole o pedaço onde você deve mudar para uma outra função e escreva testes para essa função
  - No caso de bugs, escreva um teste para a condição de falha
  - Use testes de integração

academia CHAORDIC

Exercício #3

CHaORDiC
you'll like

Implementar testes para a classe “Alarm”.

Refatorem a classe o quanto for preciso, mas evitem mudar a interface publica da classe.

Há alguns princípios de design que não são seguidos ;)

Se liga aí que agora é hora da **revisão**!

Ciclo do TDD; Boas Praticas; TDD & Design; Principios basicos de Design; Ferramental: AssertJ, Gradle Watch, Mockito; Tipos e Estilos de Teste; Mocks; Como usar TDD em código legado.

academia CHAORDIC

Como evoluir com TDD

- Seja dogmático no início
- Pratique suas habilidades
  - Bugs são um ótimo caso para TDD

academia CHAORDIC

*Always check a module in **cleaner** than when you checked it out.*

-- Uncle Bob

academia CHAORDIC

Referencias

- Exercicios
  - https://sites.google.com/site/tddproblems/
  - http://www.michael-whelan.net/code-katas-for-practicing-tdd/
- Livros
  - **Testing Driven Development by Example** - *Kent Beck*
  - **Clean Code** & **The Clean Coder** - *Robert “Uncle Bob” Martin*
  - **Working Effectively with Legacy Code** - *Michael Feathers*

academia CHAORDIC

Referencias

- Links
  - http://blog.james-carr.org/2006/11/03/tdd-anti-patterns/
  - http://devblog.avdi.org/2011/09/06/making-a-mockery-of-tdd/
  - Http://martinfowler.com/articles/mocksArentStubs.html
  - http://martinfowler.com/articles/designDead.html
  - http://martinfowler.com/articles/workflowsOfRefactoring
  - http://blog.jayfields.com/2007/06/testing-one-assertion-per-test.html
  - http://nerds.airbnb.com/testing-at-airbnb/
  - https://blog.8thlight.com/uncle-bob/2013/03/06/ThePragmaticsOfTDD.html
  - https://docs.google.com/a/chaordicsystems.com/file/d/0B_EQS8-2f7m5WENpVmo4UVF1X1k/edit
  - http://googletesting.blogspot.co.uk/2015/04/just-say-no-to-more-end-to-end-tests.html

academia CHAORDIC

CHAORDIC
you'll like

Danilo Queiroz
danilo@chaordic.com.br