tokenlab

Teste Unitário

...

Vinícius Silva - viniciussilva@tokenlab.com.br

Sumário

- Introdução
  - Testes no âmbito da Engenharia de Software
  - Testes Automatizados
- Desenvolvimento
  - Conceito de Teste Unitário e JUnit
  - Test Driven Development (TDD)
- Conclusão
  - Escrevendo testes para uma API Rest com Spring Boot

tokenlab

Testes no âmbito da Engenharia de Software

Um pouquinho de história...

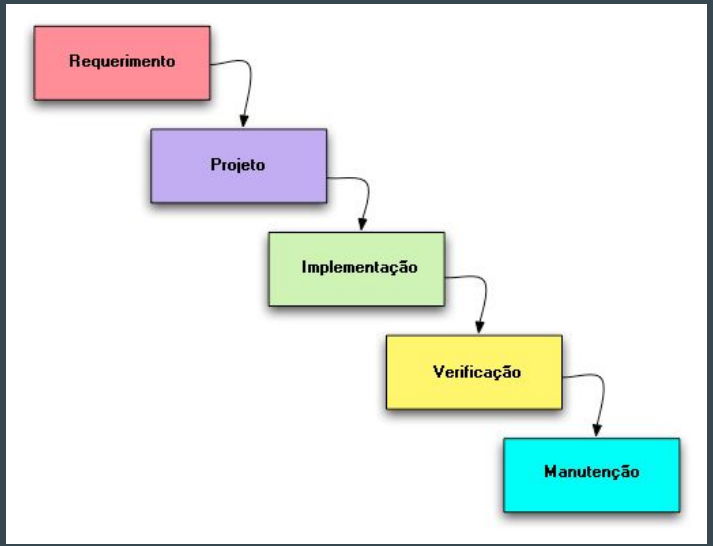
tokenlab

- Surgimento no final da década de 1960
- Conferência da OTAN, Outubro de 1968
- 12 áreas de conhecimento em Engenharia de Software
  - Testes de Software
  - Processos de Software

Processos Waterfall (ou em cascata)

tokenlab

- Propostos na década de 1970
- Inspirados em processos da Engenharia Tradicional
- Dominantes na área até a década de 1990
- Sucesso oriundo da padronização lançada pelo Departamento de Defesa Norte-Americano em 1985

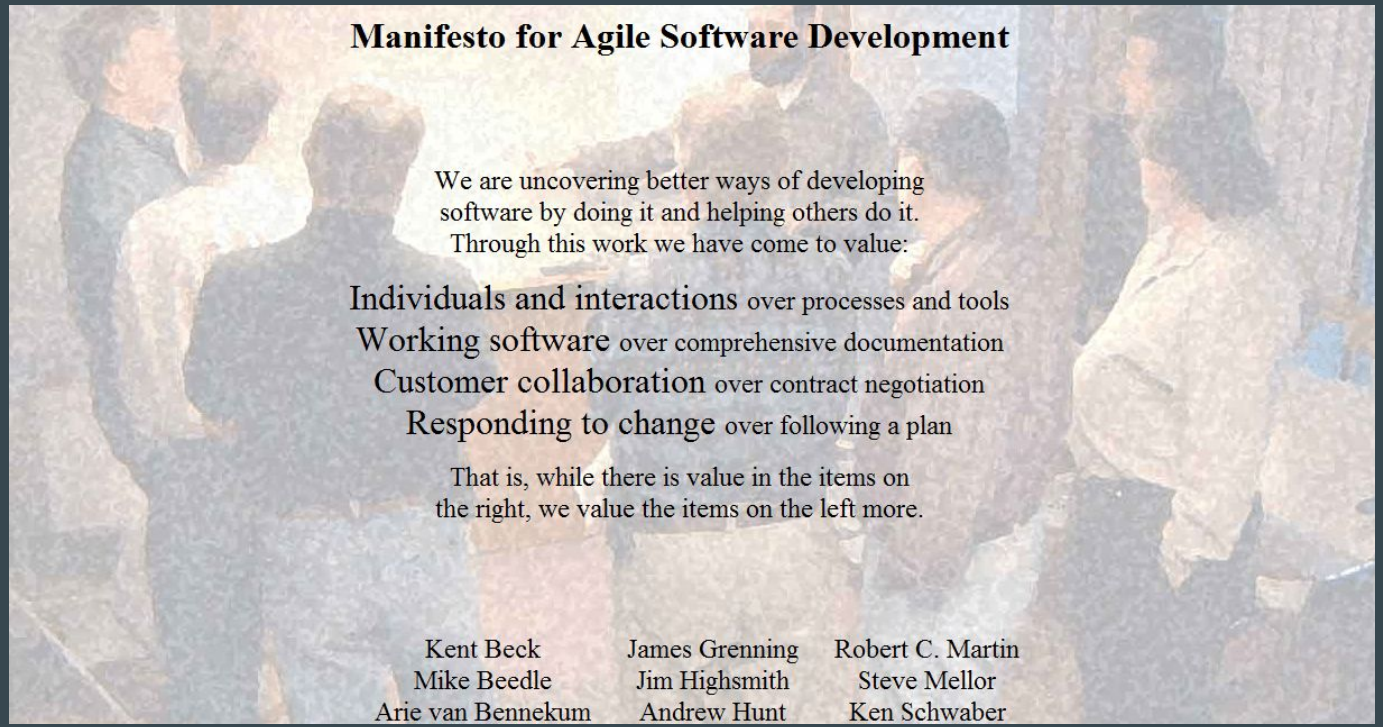


Processos Ágeis

tokenlab

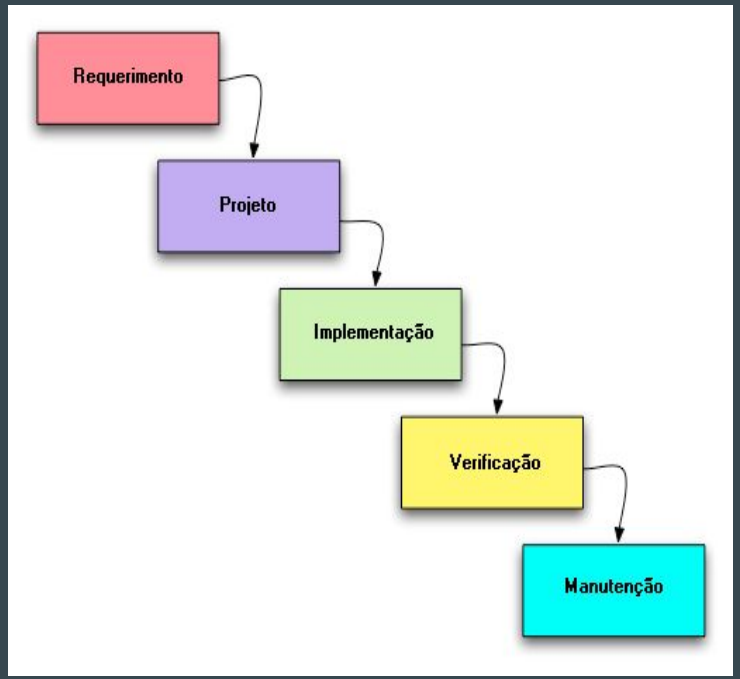
Processos Ágeis

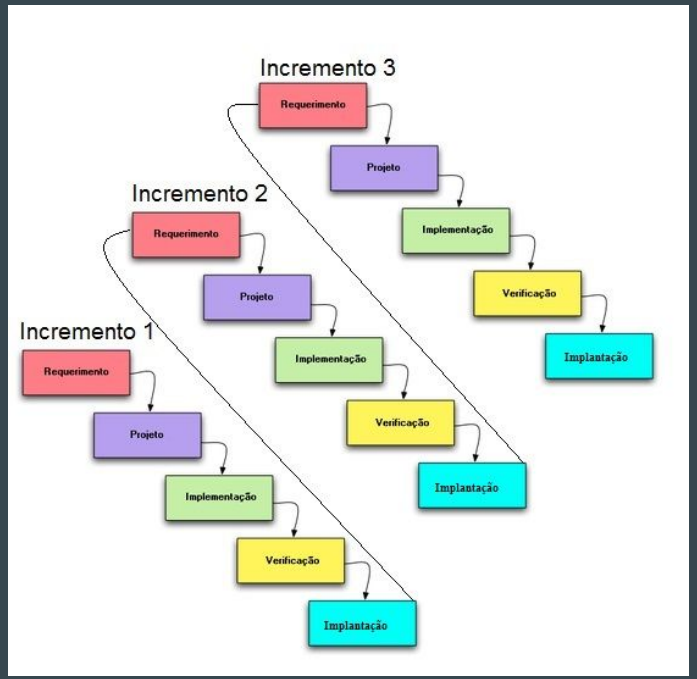
tokenlab



Processos Waterfall x Processos Ágeis

tokenlab



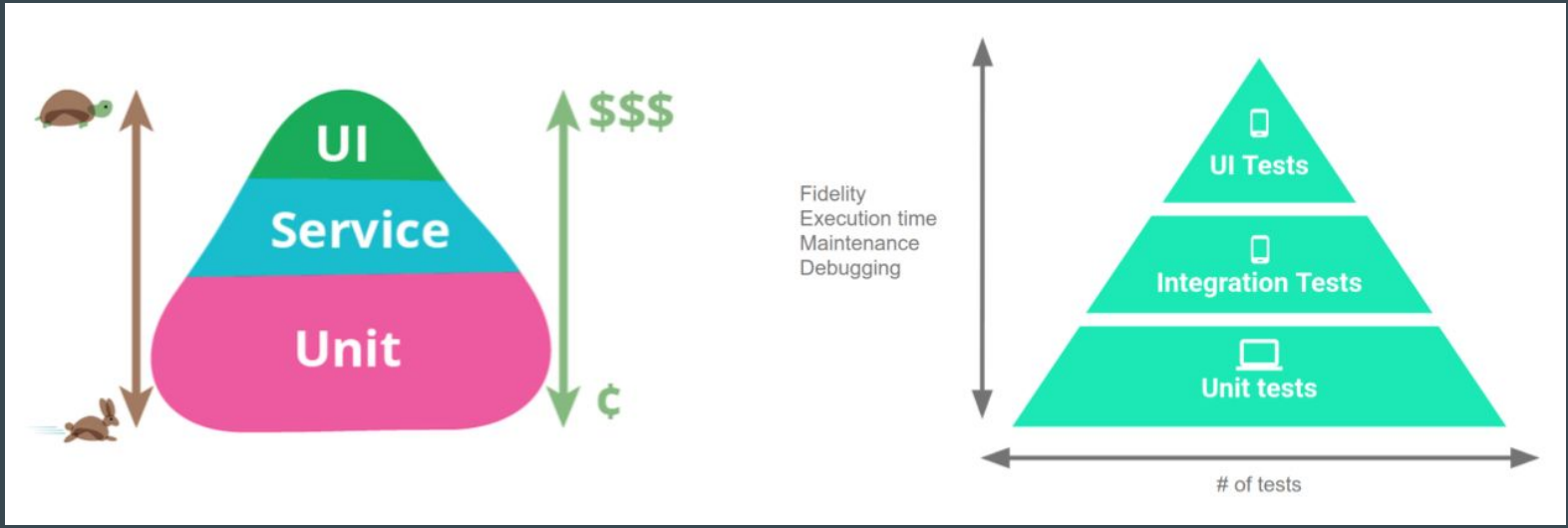


tokenlab

Testes Automatizados

Pirâmide de testes

tokenlab



Dividir para conquistar: o segredo é a amizade...

tokenlab

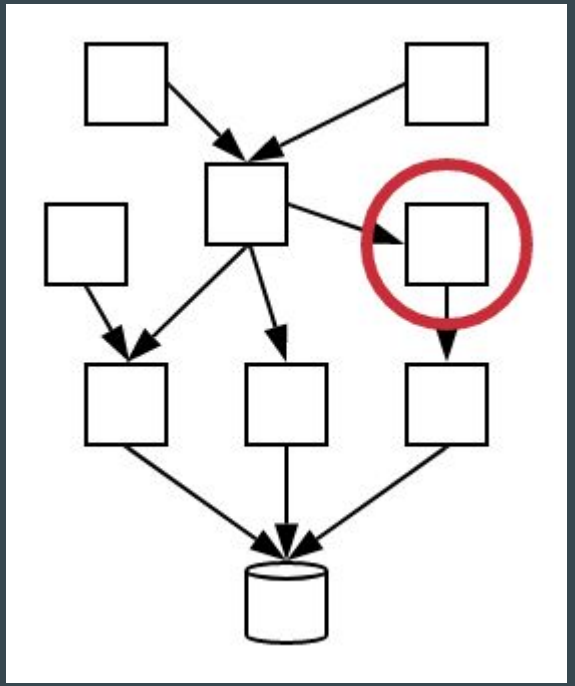
tokenlab

Conceito de Teste Unitário e JUnit

Testes de unidade

tokenlab

- Base da pirâmide
- Objectivo: verificar automaticamente pequenas partes de um código
- Desempenho: executam rapidamente
- Complexidade: baixa, fáceis de implementar

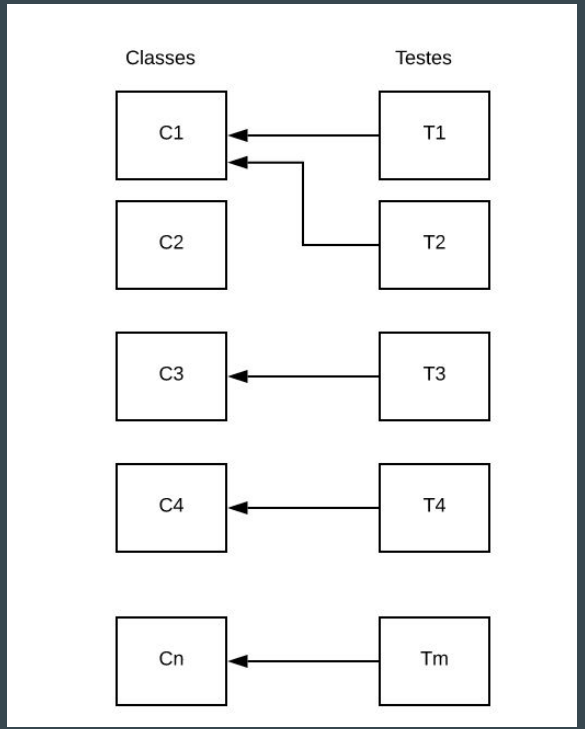


Resumindo...

tokenlab

- Um teste de unidade é um programa que chama métodos de uma classe e verifica se eles retornam os resultados esperados.

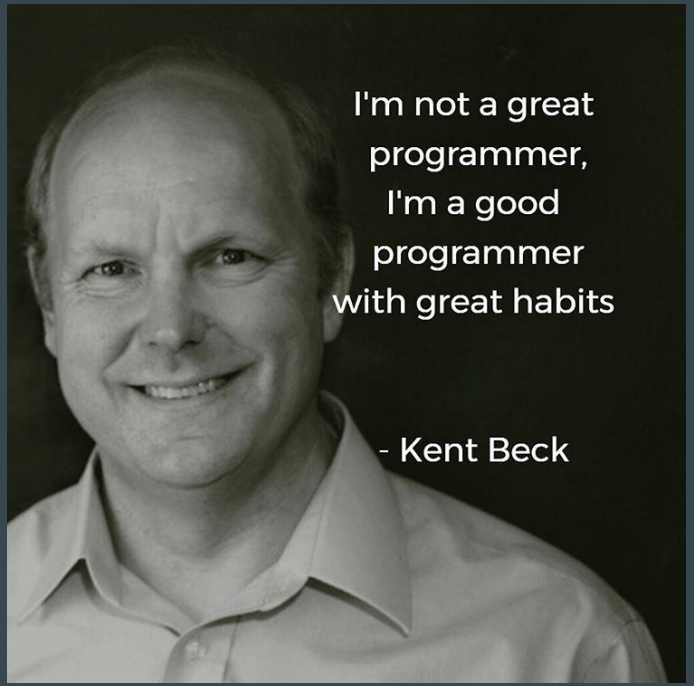
- Assim, quando se usa testes de unidades, o código de um sistema pode ser dividido em dois grupos: um conjunto de classes — que implementam os requisitos do sistema — e um conjunto de testes.



Como implementá-los?

tokenlab

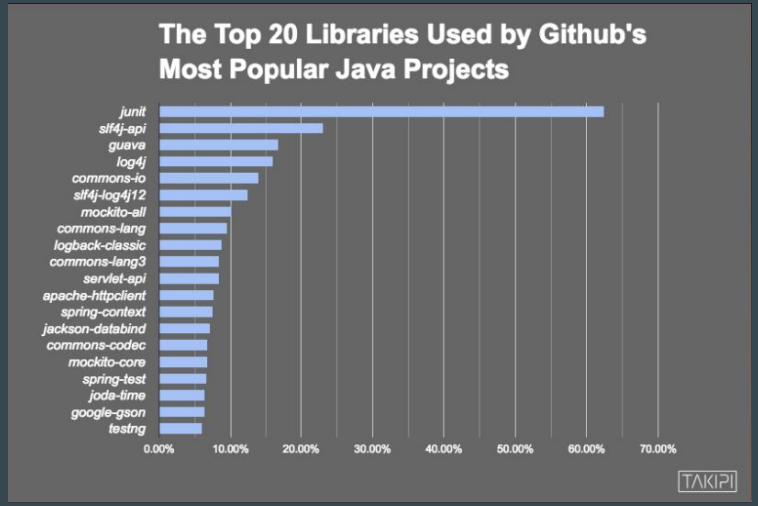
- Testes de unidade são implementados usando frameworks
- Os mais conhecidos são chamados de frameworks **xUnit**, onde o x designa a linguagem usada na implementação dos testes
- O primeiro desses frameworks foi implementado por Kent Beck no final da década de 80
- Foi implementado para a linguagem Smalltalk e seu nome era SUnit



JUnit

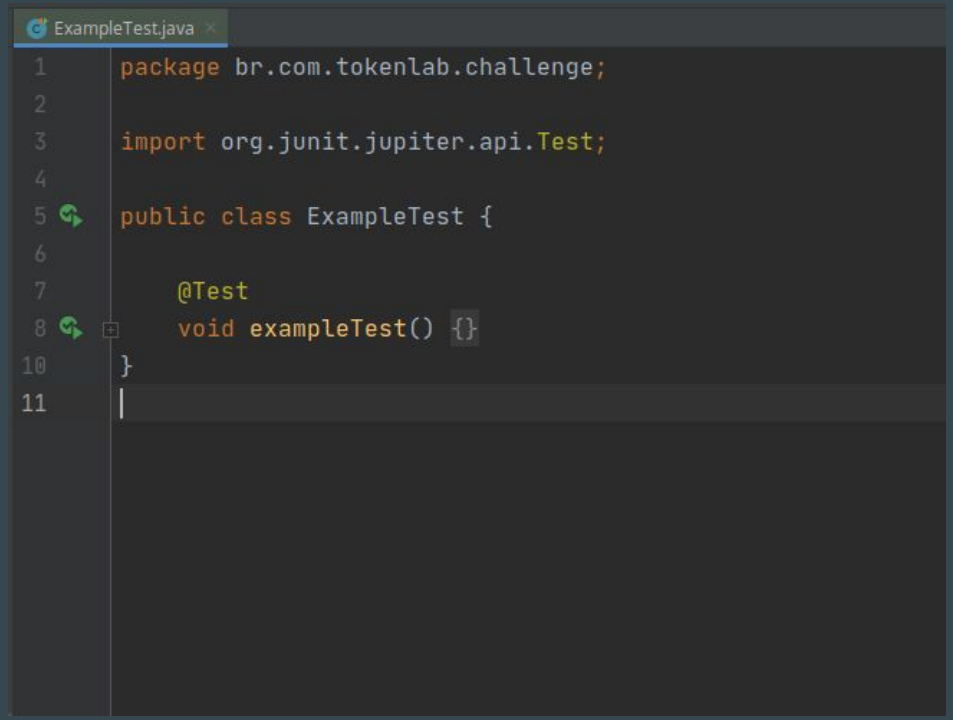
tokenlab

- A primeira versão do JUnit foi implementada em 1997
- Por Kent Beck e Erich Gamma durante uma viagem de avião entre a Suíça e os EUA
- JUnit 5 é a versão major do framework
- É o framework de testes mais popular no mundo java



E como é um teste com JUnit?!

tokenlab



JUnit Test Lifecycle

tokenlab

LifecycleTest.java

```java
package br.com.tokenlab.challenge;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.*;

public class LifecycleTest {

    @BeforeAll
    static void beforeAll() { System.out.println("LifecycleTest -> beforeAll"); }

    @BeforeEach
    void setUp() { System.out.println("LifecycleTest -> setUp"); }

    @Test
    void someTest() { System.out.println("LifecycleTest -> someTest"); }

    @Test
    void some2Test() { System.out.println("LifecycleTest -> some2Test"); }

    @AfterEach
    void tearDown() { System.out.println("LifecycleTest -> tearDown"); }

    @AfterAll
    static void afterAll() { System.out.println("LifecycleTest -> afterAll"); }
}
```

Run: LifecycleTest

Test Results
- LifecycleTest
  - some2Test()
  - someTest()

Tests passed: 2 of 2 tests – 52 ms

```
/home/viniciussilva/.sdkman/candidates/java/11.0.5-zulu/bin/java ...
LifecycleTest -> beforeAll

LifecycleTest -> setUp
LifecycleTest -> some2Test
LifecycleTest -> tearDown


LifecycleTest -> setUp
LifecycleTest -> someTest
LifecycleTest -> tearDown

LifecycleTest -> afterAll

Process finished with exit code 0
```

Exemplo: Stack e StackTest

tokenlab

```java
import java.util.ArrayList;
import java.util.EmptyStackException;

public class Stack<T> {
    private ArrayList<T> elements = new ArrayList<>();
    private int size = 0;

    public int size() {
        return size;
    }

    public boolean isEmpty() {
        return (size == 0);
    }

    public void push(T elem) {
        elements.add(elem);
        size++;
    }

    public T pop() throws EmptyStackException {
        if (isEmpty()) {
            throw new EmptyStackException();
        }
        T elem = elements.get(size-1);
        size--;
        return elem;
    }
}
```

```java
class StackTest {
    Stack<Integer> stack;

    @BeforeEach
    void setUp() { stack = new Stack<>(); }

    @Test
    public void testEmptyStack() { assertTrue(stack.isEmpty()); }

    @Test
    public void testNotEmptyStack() {
        stack.push( elem: 10);
        assertFalse(stack.isEmpty());
    }

    @Test
    public void testSizeStack() {
        stack.push( elem: 10);
        assertEquals( expected: 1,stack.size());
    }

    @Test
    public void testPushPopStack() {
        stack.push( elem: 10);
        stack.push( elem: 20);
        assertEquals( expected: 20, stack.pop());
    }

    @Test
    public void testEmptyStackException() {
        assertThrows(EmptyStackException.class, () -> stack.pop());
    }
}
```

Quando escrever um testes de unidade?

tokenlab

- Após implementar uma pequena funcionalidade
- Durante o desenvolvimento de uma funcionalidade, por exemplo, escrever algumas classes e logo em seguida seus respectivos testes
- Ao corrigir um bug, escreva um teste que o reproduz
- Trechos de código com um conjunto de ‘prints’, os quais serão deletados antes do commit.

Benefícios

- Encontrar bugs ainda na fase de desenvolvimento
- Rede de proteção contra regressões no código
- Clean code, design patterns, solid
- Documentação e especificação do código
- Novos integrantes podem começar analisando os testes para se familiarizar com o projeto

Princípios FIRST

tokenlab

- Fast: teste de unidade devem executados rapidamente, em questões de milisegundos
- Independent: a ordem de execução dos testes de unidade não é importante
- Repeatable: testes de unidade devem ter sempre o mesmo resultado, se um teste T é chamado N vezes, o resultado deve ser o mesmo nas N execuções
- Self-validating: resultado de um teste de unidades deve ser facilmente verificável
- Timely: testes de unidade devem ser escritos o quanto antes

Test Smells

- Teste obscuro: longo, complexo e difícil de entender
- Teste com lógica condicional: com comandos if, laços, etc, o ideal é que sejam lineares
- Duplicação de código em testes: código repetido em diversos métodos de teste.
- Ao identificar um test smell, deve-se refletir sobre se não é possível ter um teste mais simples e menor, com um código linear e sem duplicação de comandos

Cobertura de Testes

tokenlab

- Métrica que ajuda a definir o número de testes que precisamos escrever para um programa
- Cobertura de testes = (número de comandos executados pelos testes) / (total de comandos do programa)
- Não existe um número mágico e absoluto para cobertura de testes
- Em uma conferência de devs do Google, em 2014, algumas estatísticas sobre a cobertura de testes dos sistemas da empresa mostraram na mediana 78% de cobertura, em nível de comandos

```java
public class Stack<T> {
    private ArrayList<T> elements = new ArrayList<>();
    private int size = 0;

    public int size() {
        return size;
    }

    public boolean isEmpty() {
        return (size == 0);
    }

    public void push(T elem) {
        elements.add(elem);
        size++;
    }

    public T pop() throws EmptyStackException {
        if (isEmpty()) {
            throw new EmptyStackException();
        }
        T elem = elements.get(size-1);
        size--;
        return elem;
    }
}
```

Mocks

tokenlab

- Mock é um tipo de objeto que imita objetos reais para teste.
- Permite simular o comportamento de objetos reais complexos
- Uma chamada remota pode ser simulada com um objeto mock
- Podem ser criados através de frameworks que facilitam a sua criação
- Mockito é um exemplo de framework para criação de objetos mocks

Exemplo

tokenlab

BookSearch.java

```java
import ...

public class BookSearch {
    BookService rbs; // API REST

    public BookSearch(BookService rbs) { this.rbs = rbs; }

    public Book getBook(int isbn) throws JSONException {
        String json = rbs.search(isbn);
        JSONObject jsonObject = new JSONObject(json);
        String title = (String)jsonObject.get("title");
        return new Book(title);
    }
}
```

BookSearchTest.java

```java
import ...

class BookSearchTest {
    private BookService service;

    @BeforeEach
    void setUp() { service = new MockBookService(); }

    @Test
    public void testGetBook() throws JSONException {
        BookSearch bs = new BookSearch(service);
        String title = bs.getBook( isbn: 1234).getTitle();
        assertEquals( expected: "Eng Soft Moderna", title);
    }
}
```

BookService.java

```java
public interface BookService {
    String search(int isbn);
}
```

MockBookService.java

```java
public class MockBookService implements BookService {

    private static final String ESM = "{ \"title\": \"Eng Soft Moderna\" }";

    private static final String NULL_BOOK = "{ \"title\": \"NULL\" }";

    @Override
    public String search(int isbn) { return isbn == 1234 ? ESM : NULL_BOOK; }
}
```

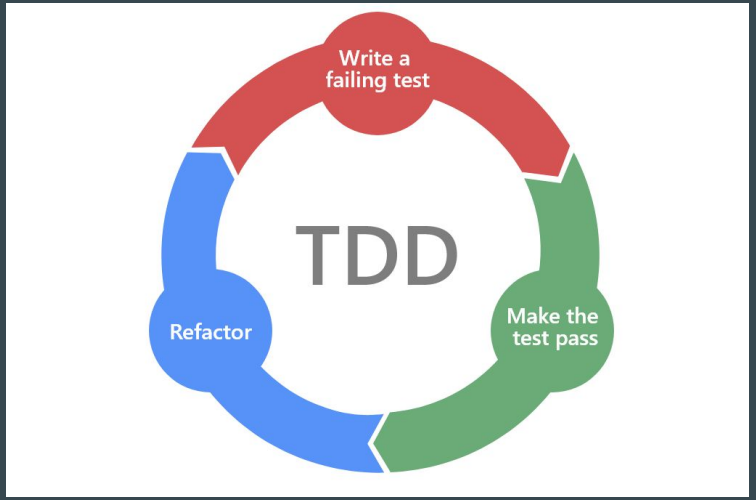
tokenlab

Test Driven Development (TDD)

Desenvolvimento Orientado a Testes

tokenlab

- É uma das práticas de programação propostas por Extreme Programming (XP)
- Nos obriga a escrever códigos menos acoplados e mais coesos para que se tornem mais “testáveis”
- É uma prática relacionada não apenas com testes, mas também com a melhoria do design de um sistema
- Ajuda a evitar que os desenvolvedores esqueçam de escrever testes



tokenlab

Escrevendo testes para uma API Rest com Spring Boot

Instruções...

tokenlab

bit.ly/3bJrVhi

Referências e dicas

- Livro Engenharia de Software Moderna: https://engsoftmoderna.info/
- Conteúdos execelentes sobre Java e Spring: https://www.baeldung.com/
- Para quem está iniciando no mundo Spring: https://www.michellibrito.com/
- Curso completo de Testes em Spring
  https://www.udemy.com/course/testing-spring-boot-beginner-to-guru/

tokenlab

Dúvidas?

tokenlab

Obrigado!

Vinícius Silva - viniciussilva@tokenlab.com.br