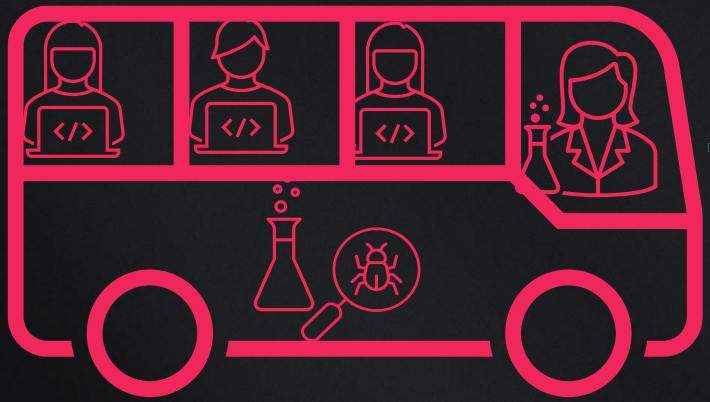
COMPLIANCE & QUALITY ASSURANCE

2024

FIAP

TDD

TEST-DRIVEN-DEVELOPMENT



CONTEÚDO DA AULA

- ESTRUTURA DO TESTE DE UNIDADE
- O QUE É TDD?
- O CICLO RED > GREEN > REFACTOR
- COMEÇANDO O TDD
- DUBLÊS DE TESTES
- BENEFÍCIOS
- PRÁTICA

OS 3 A's DO TESTE DE UNIDADE

- ARRANGE (PREPARAR): PREPARAR OBJETOS PARA O TESTE, INSTANCIAÇÃO E DADOS

- ACT (AGIR): CHAMAR A OPERAÇÃO QUE DESEJA TESTAR

- ASSERT (VALIDAR): VALIDAR O RESULTADO (RETORNO E/OU ESTADO)

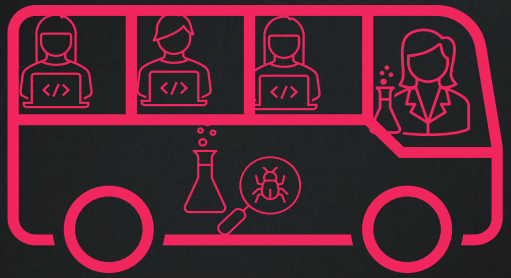
ESTRUTURA DO TESTE DE UNIDADE

```
describe("Conta", () => {
 test("depositar com sucesso", async () => {
 // Arrange
 const conta = new Conta(“123456", 5000.0);

// Act
conta.depositar(200.0);

//Assert
expect(conta.saldo).toBe(5200.0);
});
});
```

O QUE É TDD?



TÉCNICA DE DESENVOLVIMENTO DE SOFTWARE CRIADA POR KENT BECK, ONDE O DESENVOLVIMENTO DA APLICAÇÃO É GUIADO POR TESTES UNITÁRIOS ESCRITOS PREVIAMENTE.

BASEIA-SE NO CICLO RED > GREEN > REFACTOR QUE DEVE SER REPETIDO PARA CADA FUNCIONALIDADE.

É UM DOS PILARES DO XP (EXTREME PROGRAMMING).

RED > GREEN > REFACTOR

ESCREVA UM TESTE QUE FALHE

FAÇA O TESTE PASSAR

REFATORE

REPEAT!

O CICLO DEVE SER REPETIDO PARA TODAS AS FUNCIONALIDADES ESPECIFICADAS NO PROJETO

COMEÇANDO O TDD

PEGUE O REQUISITO E QUEBRE EM PEQUENOS TESTES
CODIFIQUE UM TESTE, RODE
ERRO

CODIFIQUE PARA O TESTE PASSAR, RODE
SUCESSO

MELHORE O TESTE OU CRIE OUTRO, RODE
ERRO

CODIFIQUE PARA O TESTE PASSAR, RODE
SUCESSO

FAÇA ISSO ATÉ FINALIZAR O CÓDIGO

DUBLÊS DE TESTES

MOCK: OBJETO SIMULADO QUE REPLICA O COMPORTAMENTO DO OBJETO REAL.

STUB: IMPLEMENTAÇÃO MÍNIMA DE UM OBJETO QUE FORNECE RESPOSTAS PRÉ-DEFINIDAS.

FAKE: IMPLEMENTAÇÃO SIMPLIFICADA DE UM COMPONENTE QUE SIMULA ALGUMAS FUNCIONALIDADES DE UM COMPONENTE REAL.

DUMMY: OBJETO QUE É PASSADO PARA UM MÉTODO, MAS NÃO É USADO DE FATO.

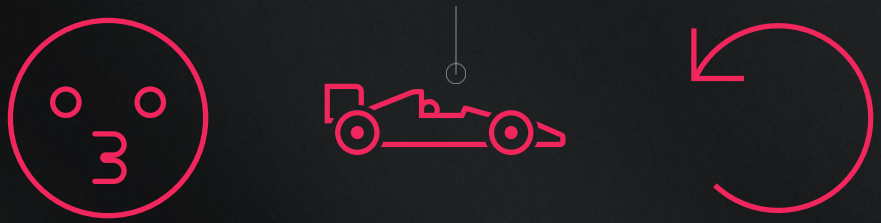
SPY: OBJETO QUE REGISTRA INFORMAÇÕES SOBRE CHAMADAS DE MÉTODOS FEITAS A OUTRO OBJETO. ASSIM VERIFICA SE CERTOS MÉTODOS FORAM CHAMADOS E COM QUAIS ARGUMENTOS.

RECOMENDAÇÕES PARA TDD?

ESCREVER TESTES SIMPLES E INCREMENTAIS.

TESTES DEVEM SER RÁPIDOS E INDEPENDENTES.

REFATORAR COM CONFIANÇA, BASEADO NOS TESTES.



O QUE NÃO É TDD?

A BALA DE PRATA DO SDLC

UMA FERRAMENTA DE TESTES DE SOFTWARE

UM MODELO DE GESTÃO DE TESTES

PERDA DE TEMPO

BENEFÍCIOS

PREVINE INSERÇÃO DE DEFEITOS NO CÓDIGO

OFERECE ALTOS ÍNDICES DE COBERTURA

PROMOVE CLEAN CODE!

FEEDBACK IMEDIATO DE QUALIDADE UNITÁRIA

SEGURANÇA PARA CORREÇÃO DE BUGS E REFATORAÇÃO

MAIOR CONFORTO PARA NOVAS IMPLEMENTAÇÕES/MANUTENÇÃO

DEVOPS FRIENDLY

DOCUMENTAÇÃO VIVA

BENEFÍCIOS

DEBUGGING
AUTOMATIZAÇÃO
FOCO NA QUALIDADE E FACILIDADE DE USO DO CÓDIGO
TESTES AUTOMATIZADOS SÃO CÓDIGO, PORTANTO DEVEM SER:

- DESACOPLADOS;
- SEM DUPLICAÇÃO;
- COESOS;
- CÓDIGO LIMPO.

QUEM FAZ TDD?

É COMUM PENSAR QUE POR SE TRATAR DA CAMADA UNITÁRIA DA APLICAÇÃO, SOMENTE DESENVOLVEDORES PODEM TRABALHAR COM TDD

NO ENTANTO, ASSIM COMO TESTES UNITÁRIOS,
QUALQUER PROFISSIONAL ESPECIALIZADO NA TÉCNICA E/OU CÓDIGO
PODE ESCREVER CASOS DE TESTE,
INCLUSIVE EM UM PROCESSO DE TDD

QUEM FAZ TDD?

É IMPORTANTE RESSALTAR QUE TDD É UMA TÉCNICA AVANÇADA DE

DESENVOLVIMENTO

E REQUER, ALÉM DE CONHECIMENTO, ENGAJAMENTO E SINERGIA DO TIME.

CUIDADO

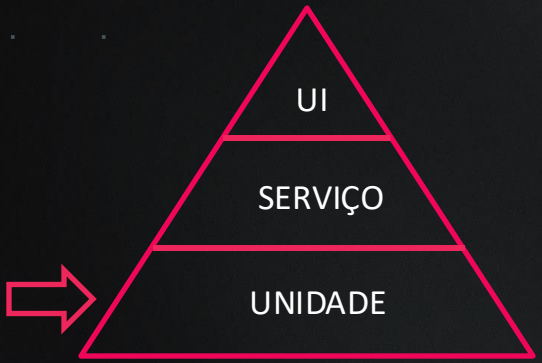
7º PRINCÍPIO DOS TESTES DE SOFTWARE:
AUSÊNCIA DE DEFEITOS É UMA ILUSÃO

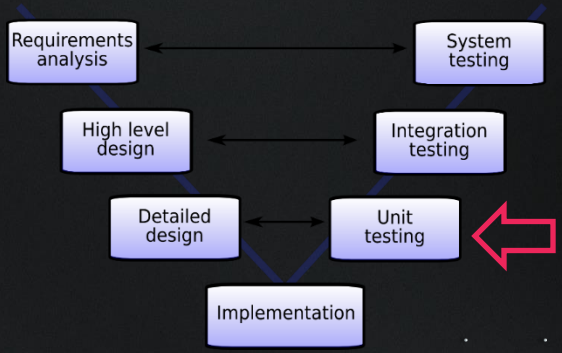
O PROCESSO DE TDD NÃO GARANTE UMA APLICAÇÃO LIVRE DE DEFEITOS.

O TDD CONSISTE EM DESENVOLVER A PARTIR DE TESTES UNITÁRIOS.

DEFEITOS AINDA PODEM SER ENCONTRADOS NAS DEMAIS
CAMADAS DA PIRÂMIDE DE TESTES OU NÍVEIS DO MODELO V.

ENTÃO DEVE COMPLEMENTAR OS OUTROS TIPOS DE TESTE E NÃO SUBSTITUÍ-LOS.

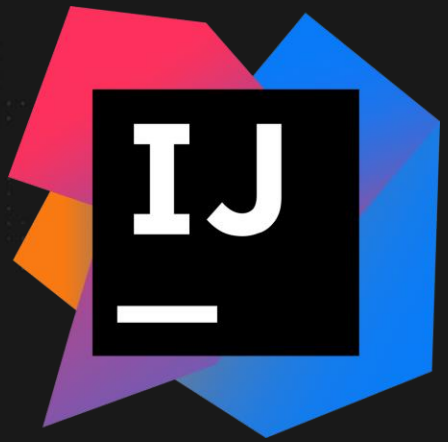

UI
SERVIÇO
UNIDADE


Requirements analysis
System testing
High level design
Integration testing
Detailed design
Unit testing
Implementation

Vamos utilizar

Java
Development Kit

Maven™



JUnit

Conceitos de Maven

Ferramenta utilizada para gerenciar projetos em Java e simplificar a vida do programador.

Auxilia no ciclo de desenvolvimento incluindo:
Compilação;
Controle de bibliotecas;
Distribuição;
Relatórios estatísticos.

O projeto nasceu a partir das dificuldades encontradas principalmente em gerenciar a compilação de projetos e no controle de bibliotecas.

Conceitos de Maven

Cada diretório de estrutura tem um propósito definido:

```
meu-projeto
├── src
│   ├── main
│   │   ├── java          (Código fonte organizado em pacotes)
│   │   ├── resources     (Recursos que a aplicação necessita)
│   │   └── webapp        (Aplicações web, web.xml, imagens, HTML)
│   └── test
│       ├── java          (Testes unitários)
│       └── resources     (Recursos que os testes necessitam)
├── target
├── pom.xml
└── README.md
```

Conceitos de Maven

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                             http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.exemplo</groupId>
    <artifactId>meu-projeto</artifactId>
    <version>1.0.0</version>
    <name>Meu Projeto</name>
    <description>Um exemplo simples de projeto Maven</description>

    <build>
        <sourceDirectory>src/main/java</sourceDirectory>
        <testSourceDirectory>src/test/java</testSourceDirectory>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <version>3.2.0</version>
                <configuration>
                    <archive>
                        <manifest>
                            <addClasspath>true</addClasspath>
                            <mainClass>com.exemplo.App</mainClass>
                        </manifest>
                    </archive>
                </configuration>
            </plugin>
        </plugins>
    </build>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

A configuração do Maven se baseia em um único arquivo, chamado **pom.xml**, que contém os metadados de um projeto.

É a partir deste arquivo que toda a mágica por trás do Maven acontece.

JUni

JUnit é um dos mais populares frameworks de teste open source para Java, que permite a criação de testes unitários automatizados.

O JUunit 5 é a versão mais recente da ferramenta e trouxe muitas mudanças em relação às versões anteriores:

- Suporte para Java 8 e versões posteriores;
- Suporte para testes parametrizados;
- Nova arquitetura que permite extensões personalizadas e suporte a diferentes motores de execução de testes;
- Melhorias de desempenho;
- Suporte para anotações de teste personalizadas;
- Suporte a testes assíncronos.

https://materiais.viaagenciadigital.com.br/guia-completo-transformacao-digital

Anatomia de um teste JUnit

Classe de Teste
Sugere-se sempre usar o mesmo nome da Classe que está sendo testada adicionando o sufixo “Test”

```
public class CalculatorTest {

    @Test
    public void testAddition() {

        Calculator calculator = new Calculator();
        int result = calculator.add(2, 3);

        assertEquals(5, result);
    }
}
```

Anotações (*Annotations*)
São usadas para definir e configurar os métodos de teste. Algumas das mais importantes são:
- @Test
- @BeforeEach / @BeforeAll
- @AfterEach / @AfterAll
- @ParameterizedTest
- @Disabled
- @Timeout

Método de Teste
Sugere-se usar um nome bastante descritivo sobre o cenário ou comportamento esperado a ser testado.

Asserções (*Assertions*)
Verificam se o resultado da lógica implementada corresponde ao esperado.
Algumas das mais importantes são:
- assertEquals() / assertNotEquals()
- assertTrue() / assertFalse()
- assertThrows() / assertDoesNotThrow()
- assertNull() / assertNotNull()

Lógica do teste
Implementações do código usado para execução do passo a passo do caso de teste.

Hands on

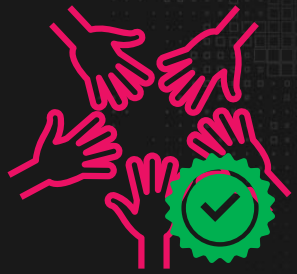
Vamos para a prática

Pré requisitos

- Instalar o JDK
- Instalar o Maven: https://maven.apache.org/download.cgi
- Instalar o IntelliJ

Hands on

Vamos para a prática



FIAP

INICIANDO O PROJETO

Hands on

Vamos para a prática

ESCREVENDO OS TESTES

Hands on

Vamos para a prática

DESENVOLVENDO A CLASSE

Alguma pergunta até aqui?

Hands on

Vamos para a prática

CRIANDO NOVOS TESTES

Exercício

Conforme fizemos em aula, uma possível implementação da Classe de teste `ApdexTest` com um método de teste para a função `calcularApdex()` ficaria assim:

ApdexTest.java

```java
package com.apdex;

import org.junit.*;
public class ApdexTest {

    @Test
    public void validarApdexExcelente(){
        //arrange
        Apdex apdex = new Apdex();
        //act
        float score = apdex.calcularApdex( s: 1000, t: 0, a: 1000);
        //assert
        Assert.assertEquals( expected: 1,score, delta: 0.001);

    }

```

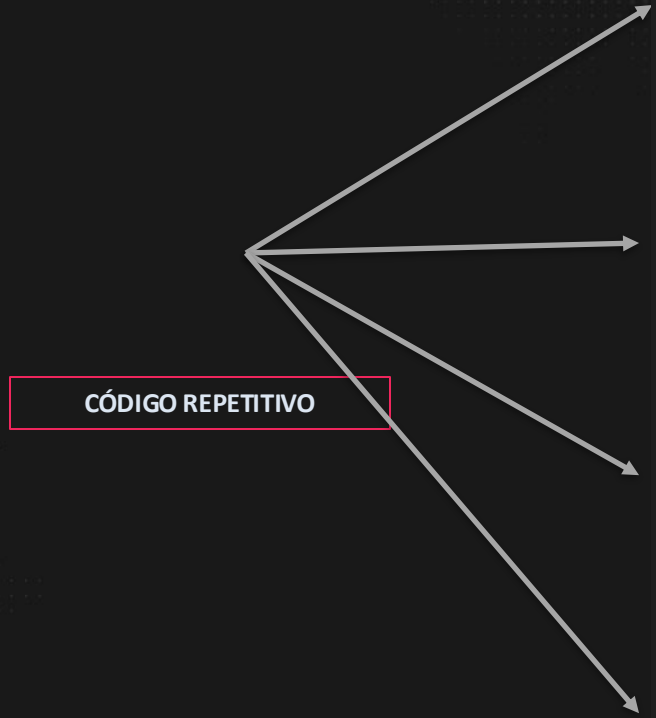
Excelente: 0,94 - 1

Agora, implemente testes unitários para os demais níveis de Apdex:

- Bom: 0,85 - 0,93
- Razoável: 0,70 - 0,84
- Ruim: 0,50 - 0,69

Inaceitável: 0,0 - 0,49

Exercício - Resolução



CÓDIGO REPETITIVO

```java
public class ApdexTest {
    @Test
    public void validarApdexExcelente(){
        //arrange
        Apdex apdex = new Apdex();
        //act
        float score = apdex.calcularApdex(1000,0,1000);
        //assert
        Assert.assertEquals(1,score,0.001);
    }
    @Test
    public void validarApdexBom(){
        //arrange
        Apdex apdex = new Apdex();
        //act
        float score = apdex.calcularApdex(870,40,1000);
        //assert
        Assert.assertEquals(0.89,score,0.001);
    }
    @Test
    public void validarApdexRazoavel(){
        //arrange
        Apdex apdex = new Apdex();
        //act
        float score = apdex.calcularApdex(770,60,1000);
        //assert
        Assert.assertEquals(0.88,score,0.001);
    }
    @Test
    public void validarApdexRuim(){
        //arrange
        Apdex apdex = new Apdex();
        //act
        float score = apdex.calcularApdex(590,0,1000);
        //assert
        Assert.assertEquals(0.59,score,0.001);
    }
    @Test
    public void validarApdexInaceitável(){
        //arrange
        Apdex apdex = new Apdex();
        //act
        float score = apdex.calcularApdex(230,460,1000);
        //assert
        Assert.assertEquals(0.46,score,0.001);
    }
```

Anotações

`@BeforeEach`	Indica que o método deve ser executado antes de cada método de teste dentro da classe atual.
`@BeforeAll`	Indica que o método deve ser executado uma única vez antes de todos os métodos de teste dentro da classe atual.
`@AfterEach`	Indica que o método deve ser executado após cada método de teste dentro da classe atual.
`@AfterAll`	Indica que o método deve ser executado antes uma única vez após todos os métodos de teste dentro da classe atual.

Hands on

Vamos para a prática

REFATORANDO OS TESTES

Exercício - Refatorando

```java
public class ApdexTest {
    2 usages
    private Apdex apdex;
    @BeforeEach
    public void arrange(){
        this.apdex = new Apdex();
    }

    @Test
    public void validarApdexExcelente(){
        //arrange
        //Apdex apdex = new Apdex();
        //act
        float score = apdex.calcularApdex( s: 1000, t: 0, a: 1000);
        //assert
        Assert.assertEquals( expected: 1,score, delta: 0.001);
    }
```

Alguma pergunta até aqui?

Assertions

As Asserções (*Assertions*) são métodos utilitários do JUnit que verificam se uma dada condição ou comportamento do código está de acordo com o que era esperado.
Esses métodos são acessados pela classe `org.junit.jupiter.api.Assertions` no JUnit 5.

- Ordem dos parâmetros:
  <esperado>, <atual>
- **Import Static**
  Por questões de legibilidade, recomenda-se o uso de importação estática da respectiva classe, para que o método seja referenciado diretamente sem o prefixo de sua classe representativa.

Assertions - Exemplos

`assertArrayEquals`	Verifica se as matrizes passadas nos parâmetros *expected* e *actual* são iguais.
`assertEquals` `assertNotEquals`	Verifica se os objetos passados nos parâmetros *expected* e *actual* são iguais ou diferentes.
`assertTrue` `assertFalse`	Verifica se dada condição retorna o booleanos Verdadeiro *True*) ou Falso (*False*).
`assertNull` `assertNotNull`	Verifica se um dado objeto é ou não nulo (*null).*
`assertThrows` `assertDoesNotThrow`	Permite verificar se um executável (*executable*) lança uma exceção do tipo especificado, ou não lança nenhuma exceção.
`assertAll`	Permite a criação de asserções agrupadas, onde todas são executadas e suas falhas reportadas em conjunto.
`fail`	Ao ser executado, atribui falha ao teste imediatamente, adicionando mensagem opcional de falha.

Alguma pergunta até aqui?

Testando *Exceptions*

Usando Try / Catch

```java
@Test
public void testExeption() {

    try {

        //Código que deve lançar Exception
        throw new IllegalArgumentException();
        fail("FALHA: Deveria ter lançado Exception, mas não lançou.");

    } catch (Exception e) {

        assertEquals(IllegalArgumentException.class, e.getClass());
        System.out.println("Exception foi gerada.");

    }
}
```

Usando *lambda expression*

```java
46    @Test
47    public void testExeption() {
48
49        assertThrows(IllegalArgumentException.class, () -> {
50            throw new IllegalArgumentException();
51        });
52
53    }
```

DESAFIO

Crie testes unitários que validem se as rotulações de apdex estão sendo atribuídas corretamente.

Implemente a rotulação na classe de Apdex conforme a tabela disponibilizada em "Exercício TDD - Cálculo de Apdex".

Referências:

https://www.devmedia.com.br/gerenciando-projetos-com-maven
https://www.devmedia.com.br/assertions-em-java
https://pt.wikipedia.org/wiki/JUnit
https://www.baeldung.com/junit-assertions
https://junit.org/junit5/docs/current/user-guide/
https://www.baeldung.com/junit-4-rules
https://bcr.bstqb.org.br/docs/syllabus_ctfl_at_2014br.pdf


FIAP