

# INF01120

## GRUPO 6

**Lojinha Dacomp**

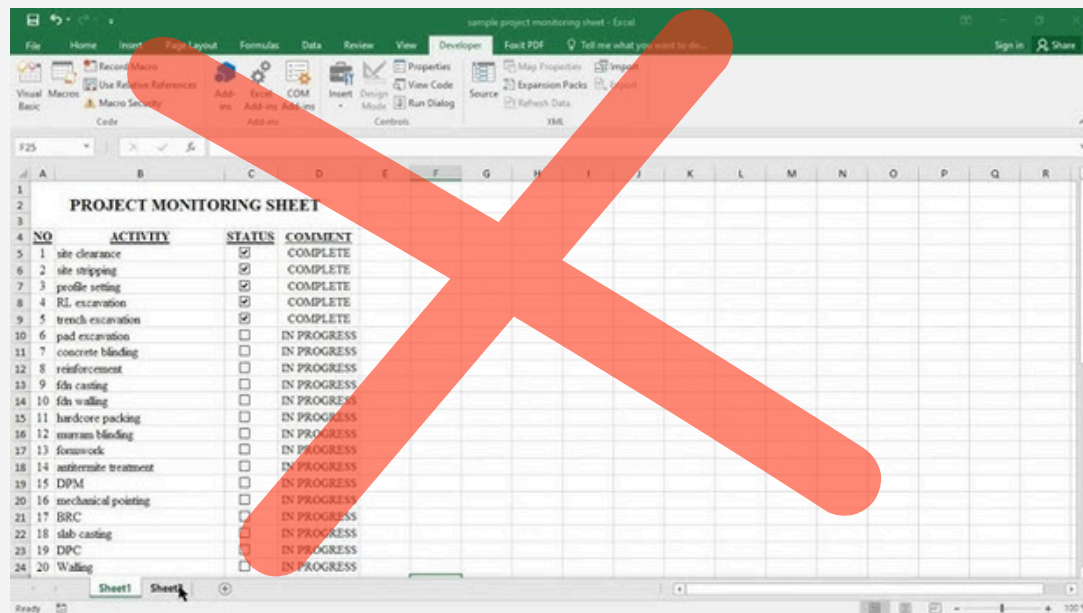
**Alunos:**

**Daniel, Diogo e Patrick**

2024.2

# PROBLEMA

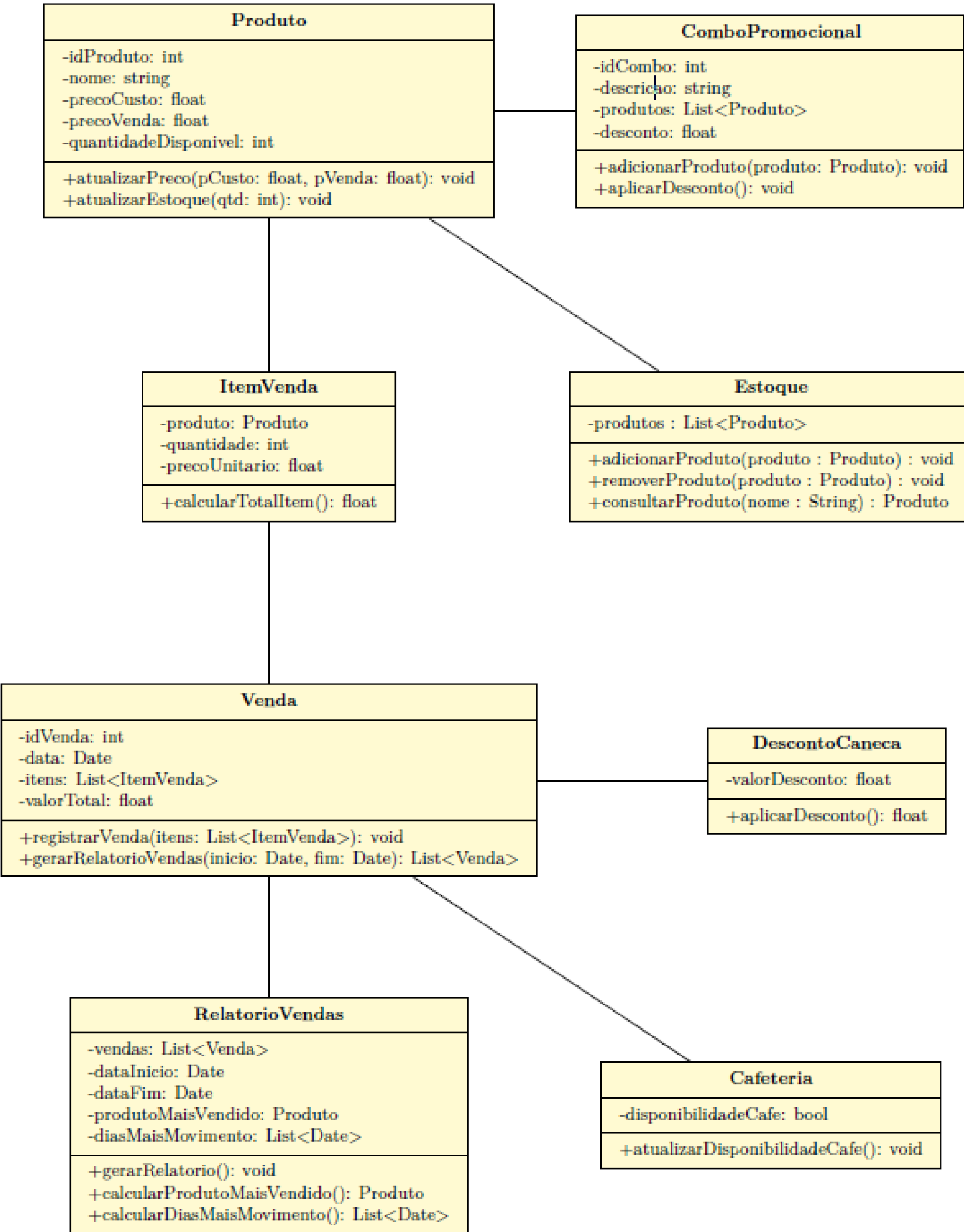
A proposta surge da necessidade de facilitar a administração da lojinha do DACOMP, centralizando a gestão de produtos, vendas e pagamento em uma única plataforma.



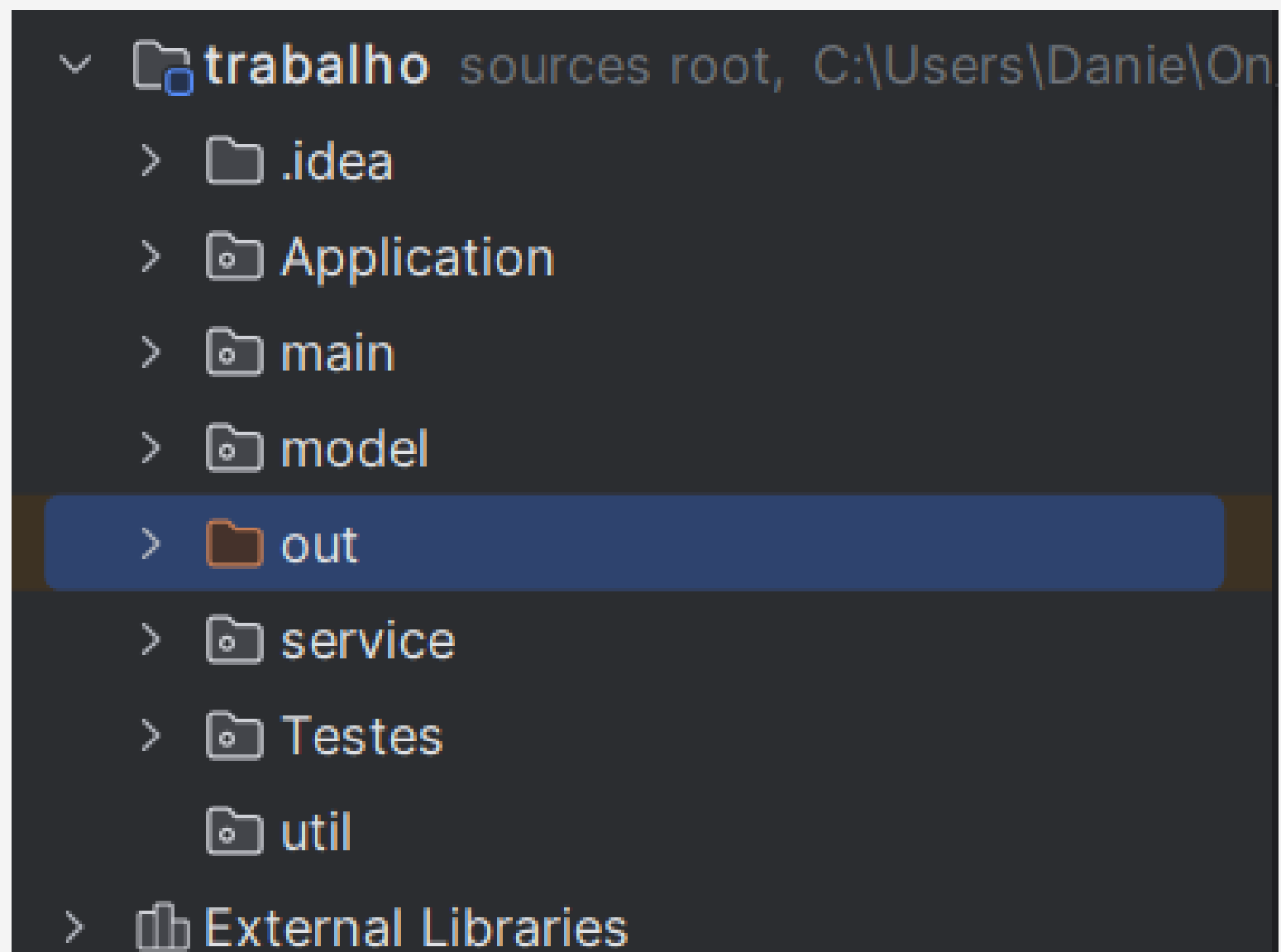
**O sistema permitirá gerenciar produtos em estoque, registrar vendas e acompanhar o histórico de transações. O software é focado no ponto de vista e necessidades do logista.**



# Diagrama de classes



# Estruturação do Projeto



**model:** Contém as classes que representam os dados e entidades principais.

- Produto, ItemVenda, Venda, ComboPromocional.

**service:** Contém as classes que implementam a lógica de negócios.

- Estoque, RelatorioVendas, DescontoCaneca, Cafeteria.

**main:** Contém a classe principal e o ponto de entrada do programa.

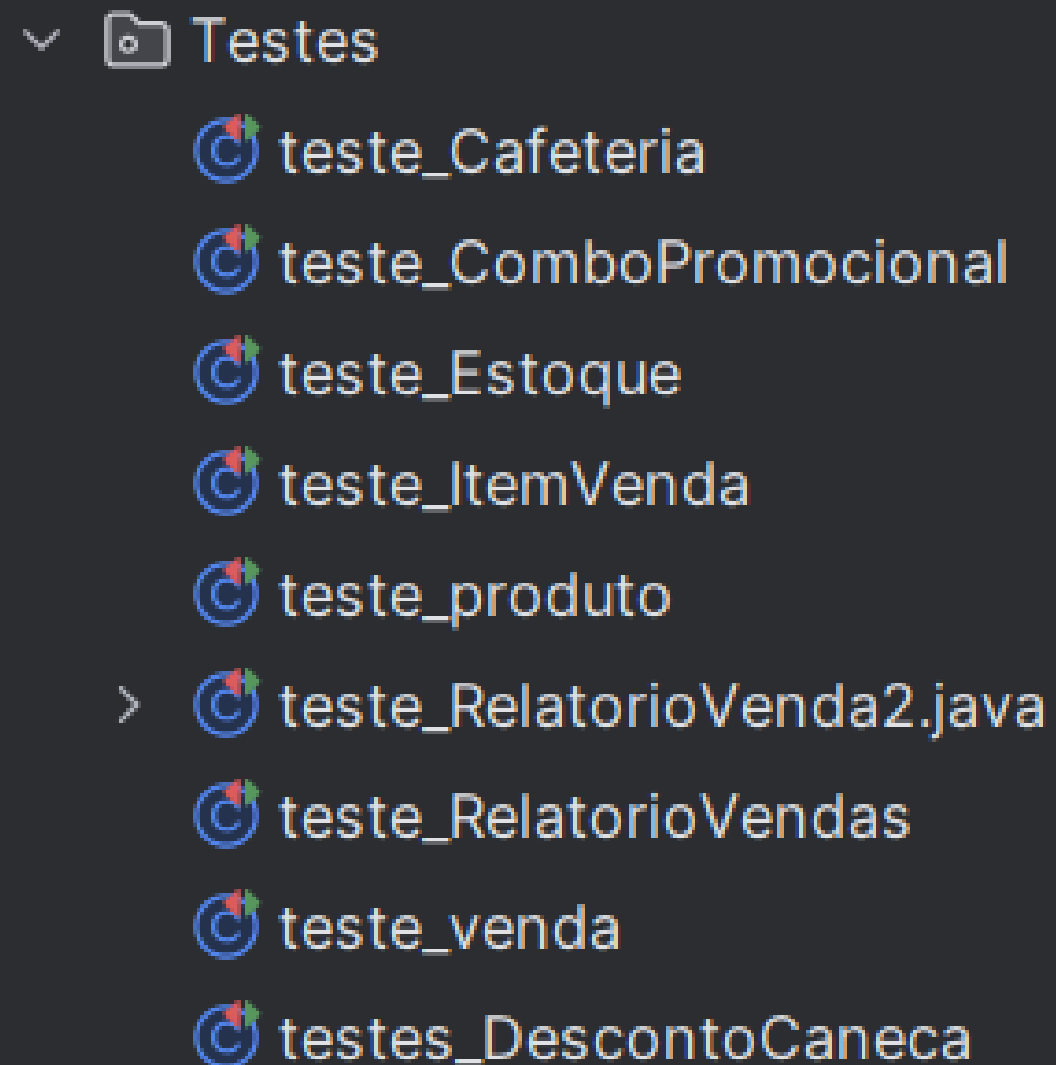
- Classe Main.

**Testes:** Contém os testes Unitários das classes.

**Application:** Contém a classe da interface Gráfica.

**util:** Contém classes auxiliares, como manipuladores de entrada/saída ou validações (se necessário).

# Testes Unitários (JUnit)



```
Testes
├── teste_Cafeteria
├── teste_ComboPromocional
├── teste_Estoque
├── teste_ItemVenda
├── teste_produto
├── > teste_RelatorioVenda2.java
├── teste_RelatorioVendas
├── teste_venda
└── testes_DescontoCaneca
```

Os testes foram implementados com o objetivo de garantir que as regras de negócios, como a adição de produtos, o processamento de vendas e a aplicação de descontos, funcionassem corretamente.

Casos de sucesso e falha: Verificação do comportamento em situações esperadas e inesperadas.

Testes de exceção: Validação se exceções são lançadas corretamente.

# Exemplos

```
// Testa a exceção ao criar um ItemVenda com quantidade zero
@Test
public void testCriarItemComQuantidadeZero() {
    assertThrows(IllegalArgumentException.class,
        () -> new ItemVenda(produto1, quantidade: 0),
        message: "Quantidade deve ser maior que zero.");
}
```

```
@Test
public void testCriacaoProdutoValido() {
    // Testa a criação de um produto com valores válidos
    Produto produto = new Produto( nome: "Café", precoCusto: 5.0, precoVenda: 10.0, quantidadeEstoque: 100);
    assertEquals( expected: "Café", produto.getNome());
    assertEquals( expected: 5.0, produto.getPrecoCusto());
    assertEquals( expected: 10.0, produto.getPrecoVenda());
    assertEquals( expected: 100, produto.getQuantidadeEstoque());
}
```

# Interface

 Sistema de Vendas

### Cadastro de Produtos

Adicionar Produto

Exibir Produtos

### Realizar Venda

Realizar Venda

Café no estoque: Disponível (100 unidades)

Verificar Disponibilidade de Café

### Criar Combo Promocional

Criar Combo Promocional

### Desconto para Canecas

Aplicar Desconto na Caneca

### Relatório de Vendas

Gerar Relatório

```
Produtos no estoque:  
Produto: banana  
Preço de Custo: R$ 10.0  
Preço de Venda: R$ 20.0  
Quantidade em Estoque: 3  
Produtos no estoque:  
Produto: banana  
Preço de Custo: R$ 10.0  
Preço de Venda: R$ 20.0  
Quantidade em Estoque: 2
```

## Console



**OBRIGADO**