Consulta de estoque - Processo de TDD

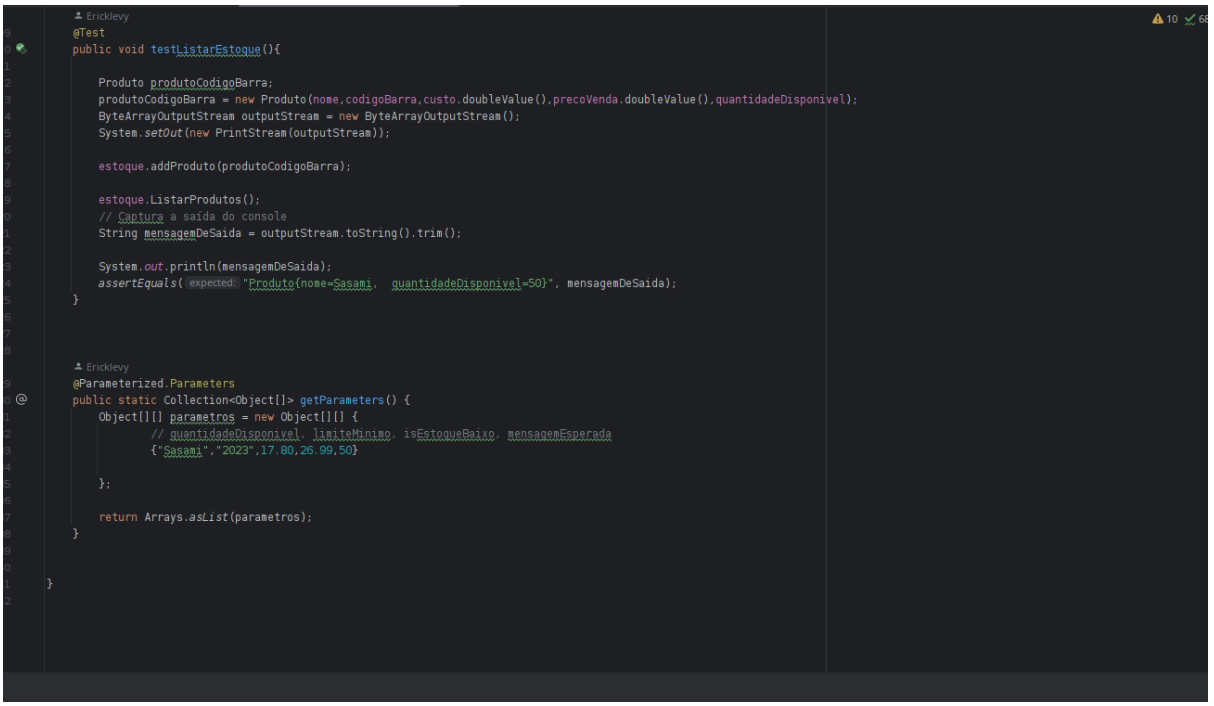
Funcionalidade : Consulta Estoque
Membro do Grupo : Erick Levy Barbosa dos Santos

github : Ericklevy

1. Na primeira etapa, realizei testes normais seguindo a abordagem TDD (Desenvolvimento Orientado por Testes) para implementar funcionalidades específicas neste programa. Utilizei a metodologia Red-Green, seguindo a lógica de escrever um teste inicial (Red), implementar a funcionalidade mínima para que o teste passe (Green), e, posteriormente, refatorar o código conforme necessário.As funções desenvolvidas incluem buscaNome, buscaCodigoBarra e listaProdutos, todas projetadas para operar com base em critérios como nome e quantidade. O processo de TDD assegurou que cada funcionalidade fosse cuidadosamente testada e validada antes de ser totalmente integrada no sistema, promovendo assim uma abordagem robusta e confiável no desenvolvimento do programa.

```java
public class Estoque {

    private List<Produto> estoqueProdutos = new ArrayList<>();

    public Produto buscaProdutoNome(String nome) {
        for (Produto produto: estoqueProdutos) {
            if(produto.getNome().equalsIgnoreCase(nome)){
                produto.buscaNome();
                return produto;
            }
        }

        return null;
    }

    public Produto buscaProdutoCodigoBarra(String codigoBarra){
        for(Produto produto: estoqueProdutos){
            if(produto.getCodigoBarra().equals(codigoBarra)){
                produto.buscaNome();
                return produto;
            }
        }

        return null;
    }


    public void addProduto(Produto produto){
        this.estoqueProdutos.add(produto);
    }

    public void ListarProdutos(){
        for(Produto produto: estoqueProdutos){
            System.out.println("Produto{" +
                    "nome=" + produto.getNome() +
                    ",  quantidadeDisponivel=" + produto.getQuantidadeDisponivel() +
                    '}');
        }
    }
```

```java
    public void setUp(){
        estoque = new Estoque();

    }



    @Test
    public void testBuscaProdutoNomeCerto(){
        Produto produtoNome;
        produtoNome = new Produto("Cartela de Ovo","2131",9.5,16.76,30);

        estoque.addProduto(produtoNome);

        Produto compara = estoque.buscaProdutoNome("Cartela de Ovo");


        assertEquals(compara.getNome(),produtoNome.getNome());
        assertEquals(compara.getCodigoBarra(),produtoNome.getCodigoBarra());
        assertEquals(compara.getCusto(),produtoNome.getCusto());
        assertEquals(compara.getPrecoVenda(),produtoNome.getPrecoVenda());
        assertEquals(compara.getQuantidadeDisponivel(),produtoNome.getQuantidadeDisponivel());

    }

    @Test
    public void testBuscaProdutoNomeErrado(){
        Produto comparaErrado = estoque.buscaProdutoNome("leite");
        assertNull(comparaErrado);



    }
```

2. Na segunda etapa, realizei a parametrização dos testes para garantir o funcionamento adequado das funções buscaNome, buscaCodigoBarra e listaProdutos. Essa abordagem envolveu a criação de testes parametrizados, nos quais diversos conjuntos de dados foram utilizados como entrada, permitindo uma cobertura mais abrangente e uma validação mais completa das funcionalidades em questão.Ao parametrizar os testes, assegurei que as funções mencionadas operassem corretamente sob diversas condições, o que contribui para a robustez e a flexibilidade do programa. Essa prática é fundamental para identificar possíveis falhas ou comportamentos inesperados em diferentes cenários de uso, garantindo a confiabilidade e a eficiência das funcionalidades implementadas.
3. 

```
Ericklevy
@Test
public void testListarEstoque(){

    Produto produtoCodigoBarra;
    produtoCodigoBarra = new Produto(nome,codigoBarra,custo.doubleValue(),precoVenda.doubleValue(),quantidadeDisponivel);
    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    System.setOut(new PrintStream(outputStream));

    estoque.addProduto(produtoCodigoBarra);

    estoque.ListarProdutos();
    // Captura a saída do console
    String mensagemDeSaida = outputStream.toString().trim();

    System.out.println(mensagemDeSaida);
    assertEquals( expected: "Produto{nome=Sasami,  quantidadeDisponivel=50}", mensagemDeSaida);
}



Ericklevy
@Parameterized.Parameters
public static Collection<Object[]> getParameters() {
    Object[][] parametros = new Object[][] {
            // quantidadeDisponivel, limiteMinimo, isEstoqueBaixo, mensagemEsperada
            {"Sasami","2023",17.80,26.99,50}

    };

    return Arrays.asList(parametros);
}


}
```

4.

```
Ericklevy
@Test
public void testBuscaProdutoNomeErrado(){
    Produto comparaErrado = estoque.buscaProdutoNome("leite");
    assertNull(comparaErrado);



}

Ericklevy +1
@Test
public void testBuscaProdutoCodigoBarraCerto(){
    //teste para o caso certo
    Produto produtoCodigoBarra;
    produtoCodigoBarra = new Produto(nome,codigoBarra,custo.doubleValue(),precoVenda.doubleValue(),quantidadeDisponivel);
    estoque.addProduto(produtoCodigoBarra);
    Produto compara = estoque.buscaProdutoCodigoBarra("2023");;



    assertEquals(compara.getNome(),produtoCodigoBarra.getNome());
    assertEquals(compara.getCodigoBarras(),produtoCodigoBarra.getCodigoBarras());
    assertEquals(compara.getPrecoCompra(),produtoCodigoBarra.getPrecoCompra());
    assertEquals(compara.getPrecoVenda(),produtoCodigoBarra.getPrecoVenda());
    assertEquals(compara.getQuantidadeDisponivel(),produtoCodigoBarra.getQuantidadeDisponivel());


}



Ericklevy
@Test
public void testBuscaProdutoCodigoBarraErrado(){

    //teste para o caso errado
    Produto comparaErrado = estoque.buscaProdutoCodigoBarra("2000");
    assertNull(comparaErrado);
}
```

5.

```
Ericklevy
@Before
public void setUp(){
    estoque = new Estoque();

}

no usages   Ericklevy
public TestesConsultaEstoque (String nome, String codigoBarra, double custo, double precoVenda, Integer quantidadeDisponivel ){
    this.nome = nome;
    this.codigoBarra = codigoBarra;
    this.custo = new BigDecimal(custo);
    this.precoVenda = new BigDecimal(precoVenda);
    this.quantidadeDisponivel = quantidadeDisponivel;
}



Ericklevy +1
@Test
public void testBuscaProdutoNomeCerto(){
    Produto produtoNome;
    produtoNome = new Produto(nome,codigoBarra,custo.doubleValue(),precoVenda.doubleValue(),quantidadeDisponivel);

    estoque.addProduto(produtoNome);

    Produto compara = estoque.buscaProdutoNome("Sasami");


    //teste

    assertEquals(compara.getNome(),produtoNome.getNome());
    assertEquals(compara.getCodigoBarras(),produtoNome.getCodigoBarras());
    assertEquals(compara.getPrecoCompra(),produtoNome.getPrecoCompra());
    assertEquals(compara.getPrecoVenda(),produtoNome.getPrecoVenda());
    assertEquals(compara.getQuantidadeDisponivel(),produtoNome.getQuantidadeDisponivel());
}

Ericklevy
```