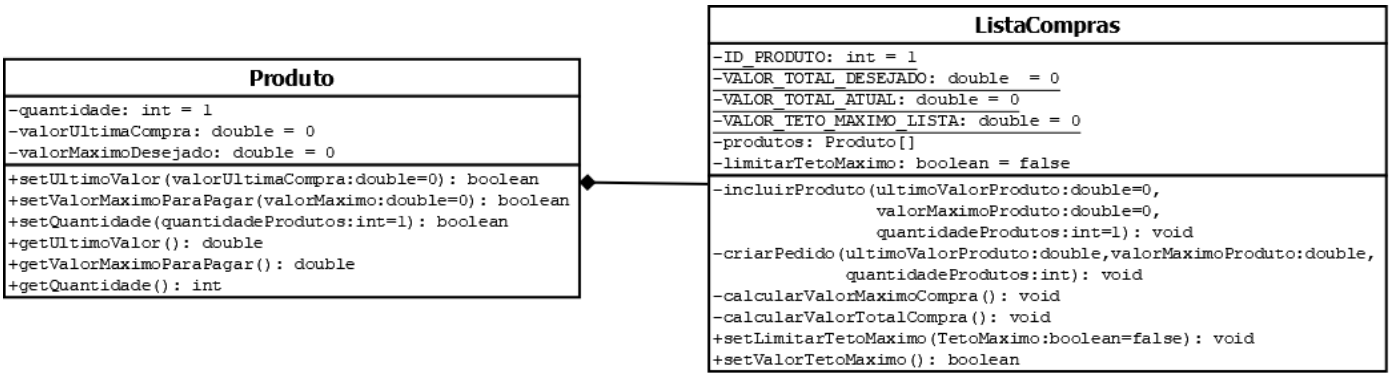
Exercício 1 e 2



Exercício 3

Método para incluir um produto no planejamento de compras:

```java
    /**
     * Inclui novos produtos no Array de objetos Produto[] com capacidade de ate 50
     * produtos.
     *
     * @param ultimoValorProduto valor do produto na ultima compra.
     * @param valorMaximoProduto valor maximo que ira pagar no produto nessa compra.
     * @param quantidadeProdutos quantidade de produtos.
     * @return
     *         Retorna int 1 se a lista nao tem teto maximo definido.
     *         Retorna int 2 se a lista tem teto maximo definido e o preco do valor
     *         total atual ainda não ultrapassou o teto mais de uma vez.
     *         Retorna int 3 se a lista tem teto maximo definido e o preco do valor
     *         total atual ja ultrassou o teto mais de uma vez.
     */
    public int incluirProduto(double ultimoValorProduto, double valorMaximoProduto, int quantidadeProdutos) {

        if (this.limitarTetoMaximo) {

            if (VALOR_TOTAL_ATUAL <= VALOR_TETO_MAXIMO_LISTA) {

                criarProdutos(ultimoValorProduto, valorMaximoProduto, quantidadeProdutos);

                return 2;

            } else {

                return 3;

            }

        } else {

            criarProdutos(ultimoValorProduto, valorMaximoProduto, quantidadeProdutos);

            return 1;

        }

    }
```

Método criarProdutos:

```java
public void criarProdutos(double ultimoValorProduto, double valorMaximoProduto, int quantidadeProdutos) {
    produtos[ID_PRODUTO] = new Produto();

    this.produtos[ID_PRODUTO].setQuantidade(quantidadeProdutos);
    this.produtos[ID_PRODUTO].setUltimoValor(ultimoValorProduto);
    this.produtos[ID_PRODUTO].setValorMaximoParaPagar(valorMaximoProduto);

    calcularValorMaximoEstimado();
    calcularValorTotalCompra();

    ID_PRODUTO++;
}
```

Teste da classe produto:

```java
public class ProdutoTest {

    @Test
    void testGet_E_SetQuantidade() {

        Produto produto = new Produto();

        assertTrue(produto.setQuantidade(quantidadeProdutos: 2));
        assertEquals(2, produto.getQuantidade());

        assertFalse(produto.setQuantidade(quantidadeProdutos: 0));
        assertEquals(1, produto.getQuantidade());

        assertTrue(produto.setQuantidade(quantidadeProdutos: 1));
        assertEquals(1, produto.getQuantidade());

    }

    @Test
    void testGet_E_SetUltimoValor() {
        Produto produto = new Produto();

        assertTrue(produto.setUltimoValor(ultimoValorCompra: 12));
        assertEquals(12, produto.getUltimoValor());

        assertTrue(produto.setUltimoValor(ultimoValorCompra: 0));
        assertEquals(0, produto.getUltimoValor());

        assertFalse(produto.setUltimoValor(-2));
        assertEquals(0, produto.getUltimoValor());



    }

    @Test
    void testGet_E_SetValorMaximoParaPagar() {
        Produto produto = new Produto();

        assertTrue(produto.setValorMaximoParaPagar(valorMaximoDesejado: 120));
```

```java
    @Test
    void testGet_E_SetValorMaximoParaPagar() {
        Produto produto = new Produto();

        assertTrue(produto.setValorMaximoParaPagar(valorMaximoDesejado: 120));
        assertEquals(120, produt
                          double Produto.getValorMaximoParaPagar()
                          Retorna o valor de valorMaximoDesejado
        assertTrue(produto.setVa                                  ;
        assertEquals(0, produto.getValorMaximoParaPagar());

        assertFalse(produto.setValorMaximoParaPagar(-1));
        assertEquals(0, produto.getValorMaximoParaPagar());
    }

}
```

Repositório github: https://github.com/luizhenrique055/Atividades-POO/tree/main/ET1