O código apresentado implementa um sistema de vendas em C# que gerencia produtos, clientes e pedidos dentro de uma loja. Esse sistema é projetado para lidar tanto com produtos físicos quanto com produtos digitais, permitindo que clientes façam pedidos, adicionem e removam produtos, e calculem o valor total do pedido. Além disso, há um controle de estoque, garantindo que os produtos só possam ser adicionados ao pedido se houver unidades disponíveis.

A estrutura do código é dividida em várias classes: `Produto`, `ProdutoFisico`, `ProdutoDigital`, `Cliente`, `Pedido` e `Loja`. Cada uma delas tem uma responsabilidade específica dentro do sistema, tornando o código modular, reutilizável e de fácil entendimento. A classe `Produto` é a base para todos os produtos disponíveis na loja, definida como uma classe abstrata (não podendo ser instanciada diretamente). Ela serve como modelo para os produtos, definindo as principais propriedades, como nome, código, preço e estoque. O construtor da classe inicializa essas propriedades, garantindo que, ao criar um produto (físico ou digital), todas as informações necessárias sejam fornecidas. A classe também possui o método abstrato `CalcularPrecoFinal`, que deve ser implementado nas subclasses `ProdutoFisico` e `ProdutoDigital`, visto que o cálculo do preço final varia de acordo com o tipo de produto.

A classe `ProdutoFisico` herda de `Produto` e representa os produtos que possuem peso e categoria, como "eletrônicos" ou "móveis". Ela acrescenta as propriedades peso (usado para calcular o custo de envio) e categoria (para classificar o produto). O método `CalcularPrecoFinal` é sobrescrito, calculando o preço final somando o preço do produto, a taxa de imposto (10% sobre o preço base) e o custo de envio (calculado com base no peso, multiplicado por 2). Já a classe `ProdutoDigital`, também herdeira de `Produto`, representa produtos digitais, como softwares, ou cursos online, com as propriedades tamanhoArquivo e formato. Seu método `CalcularPrecoFinal` aplica um desconto de 10% ao preço, refletindo a ausência de custos de produção e envio, característicos dos produtos físicos.

A classe `Cliente` representa os clientes da loja e possui as propriedades nome e numeroIdentificacao. Ao instanciar um objeto dessa classe, os dados do cliente são passados como parâmetros e armazenados nas propriedades correspondentes. Já a classe `Pedido` é onde o processo de compra acontece, gerenciando os produtos que foram adicionados ao pedido, calculando o total a ser pago e controlando o status do pedido. O pedido contém as propriedades cliente (representando o cliente que fez o pedido), dataPedido (a data de criação do pedido), produtos (a lista de produtos no pedido) e status (que começa como "Em Processamento"). Os métodos dessa classe permitem adicionar produtos ao pedido (caso haja estoque), remover

produtos (repondo o estoque), calcular o total do pedido (somando os preços finais dos produtos) e finalizar o pedido (alterando o status para "Concluído").

A classe `Loja` é responsável por gerenciar o cadastro de produtos e clientes. Ela mantém listas dos produtos e clientes cadastrados e oferece métodos para adicionar novos produtos e clientes, bem como consultar produtos por código e clientes por número de identificação. Essa centralização permite um gerenciamento eficiente dos dados da loja.

Na classe principal `Program`, o sistema começa criando uma loja e cadastrando produtos e clientes. São criados três produtos físicos (um iPhone, um notebook Dell e um monitor Samsung) e três produtos digitais (um jogo, um software e um curso de C#), que são adicionados à lista de produtos da loja. Em seguida, dois clientes são cadastrados. Cada cliente realiza um pedido, adicionando e removendo produtos, e o sistema calcula o total a ser pago por cada pedido com base nos preços finais dos produtos adicionados. Por exemplo, o cliente Marcelo adiciona um notebook, um monitor e um curso de C# ao seu pedido, depois remove o monitor e finaliza o pedido, com o sistema calculando o total correspondente.

Esse código exemplifica como modelar um sistema de vendas em C# utilizando conceitos importantes da orientação a objetos, como herança, polimorfismo e encapsulamento. A modularidade do código facilita sua manutenção e permite futuras expansões, como a inclusão de novos tipos de produtos ou métodos de cálculo de preço. Além disso, o controle de estoque garante que apenas produtos disponíveis sejam adicionados ao pedido, evitando a venda de itens esgotados.