1. Criando o estoque
Estoque estoque = new Estoque(); // Cria um objeto Estoque vazio
O que acontece: Instancia a classe Estoque, criando a lista produtos dentro dela.
Conceito: Objeto - instância de uma classe; Construtor - método especial que inicializa o objeto.

2. Estrutura de repetição para o menu
do { ... } while (true);
O que acontece: Cria um loop infinito que mantém o menu ativo até o usuário escolher sair.
Conceito: do-while executa o bloco de código pelo menos uma vez e repete enquanto a condição for verdadeira.

3. Exibindo o menu
print("\nMenu:"); print("\n1. Adicionar Produto"); ... print("\nEscolha uma opção: ");
O que acontece: Mostra as opções para o usuário.
Função print: Abreviação para Console.Write, sem quebrar linha, para manter o texto contínuo.
Conceito: Separar entrada/saída facilita manutenção do código.

4. Lendo a opção do usuário
string opcao = read(); Console.Clear();
O que acontece: read() lê o que o usuário digitou no console.
Console.Clear() limpa a tela antes de executar a ação escolhida.
Conceito: Entrada (Console.ReadLine()) e saída (Console.WriteLine()) são essenciais para interação com o usuário.

5. Estrutura de decisão: switch-case
switch (opcao) { case "1": ... break; case "2": ... break; ... }
O que acontece: Avalia a opção escolhida. Executa o bloco correspondente. break interrompe o case.
Conceito: switch é mais organizado que vários if-else quando lidamos com múltiplas opções fixas.

6. Caso 1: Adicionar produto
print("Digite o ID do produto: "); int id = int.Parse(read());
O que acontece: Solicita o ID do produto e converte o texto digitado em número inteiro.
Conceito: int.Parse converte string para int.

if (estoque.buscarProdutoPorId(id) != null) { print("\nID já existe..."); read(); continue; }
O que acontece: Verifica se já existe produto com o mesmo ID.
Conceito: Evita duplicidade usando validação antes de criar o produto. continue pula para a próxima iteração do loop.

Produto novoProduto = new Produto(id, read("Digite o nome do produto: "), double.Parse(read("Digite o preço do produto: ")), int.Parse(read("Digite a quantidade do produto: ")));
O que acontece: Cria um novo objeto Produto com todos os atributos preenchidos.
Conceitos importantes: Construtor completo; double.Parse converte texto para número decimal.

estoque.adicionarProduto(novoProduto); print("Produto adicionado com sucesso!"); read(); Console.Clear();
Adiciona à lista do estoque e confirma para o usuário.

7. Caso 2: Remover produto
int idRemover = int.Parse(read("Digite o ID do produto a ser removido: "));
if (estoque.removerProduto(idRemover)) { print("Produto removido com sucesso!"); }
Busca o produto pelo ID e remove da lista. Retorna true se removido, false se não encontrado.

8. Caso 3: Buscar produto por ID
int idBuscar = int.Parse(read("Digite o ID do produto a ser buscado: "));
Produto produtoBuscado = estoque.buscarProdutoPorId(idBuscar);
if (produtoBuscado != null) { produtoBuscado.mostrar(); } else { print("Produto não encontrado."); }
Explicação: Usa LINQ (FirstOrDefault) para localizar produto. Se encontrado, chama o método mostrar() para exibir informações.

9. Caso 4: Adicionar quantidade a um produto
int idAdicionar = int.Parse(read("Digite o ID do produto: "));
int quantidadeAdicionar = int.Parse(read("Digite a quantidade a ser adicionada: "));
if (estoque.addQuantidadeProduto(idAdicionar, quantidadeAdicionar)) { print("Quantidade adicionada com sucesso!"); }
O que acontece: Localiza o produto e soma a quantidade informada. Conceito: Reaproveitamento de métodos (addQuantidadeProduto) da classe Estoque.

10. Caso 5: Mostrar produtos
string filtro = read("Digite um filtro para buscar produtos (deixe em branco para mostrar todos): ");
estoque.mostrarProdutos(filtro);
Explicação: Pode exibir todos os produtos ou apenas aqueles cujo nome contém o texto digitado.
Conceito: Filtragem usando LINQ: Where(p => p.nome.Contains(filtro)).

11. Caso 6: Apagar lista de produtos
estoque.apagarListaProdutos();
O que acontece: Limpa toda a lista de produtos. Conceito: List.Clear() remove todos os elementos de uma lista.

12. Caso 7: Calcular valor total do estoque
estoque.calcularValorTotalEstoque();
Explicação: Soma preco * quantidade de todos os produtos usando LINQ Sum.

13. Caso 8: Sair
print("Saindo..."); return;
O que acontece: Encerra o programa. Conceito: return em Main finaliza a execução do programa.

14. Caso 9: Diminuir quantidade de um produto
int idDiminuir = int.Parse(read("Digite o ID do produto: "));
int quantidadeDiminuir = int.Parse(read("Digite a quantidade a ser diminuída: "));
if (estoque.subQuantidadeProduto(idDiminuir, quantidadeDiminuir)) { print("Quantidade diminuída com sucesso!"); }
O que acontece: Localiza o produto e subtrai a quantidade informada. Validação: O método remover da classe Produto impede que o estoque fique negativo.