Aluna: Isis Yasmim Almeida de Sousa

Trabalho Prático: OSLite (Classes/Records/Structs e Enums, com Navegabilidade Bidirecional)

O uso de records/structs e enums teve um impacto significativo na clareza, testabilidade e segurança do domínio.

Entidades como Cliente e OrdemDeServico foram modeladas como **classes** para enfatizar sua **identidade** única e mutabilidade controlada (ex.: mudança de StatusOS ou adição de ItemDeServico). Em contraste, Money foi implementada como um **record struct**, representando a **semântica de valor**; sua imutabilidade garante que, ao usar Money, seu valor não mudará inesperadamente, prevenindo erros e facilitando testes de igualdade por valor (se dois objetos Money valem o mesmo, são considerados iguais, independentemente da referência). Essa diferenciação (identidade vs. valor) deixa claro o papel de cada objeto.

StatusOS como **enum** explicitou os estados válidos, eliminando "strings mágicas" e tornando as regras de transição (ex.: IniciarExecucao exige StatusOS.Aberta) óbvias e fortemente tipadas. Isso melhorou a legibilidade e permitiu que a validação fosse ***fail-fast*** (lançando InvalidOperationException), o que simplificou o TDD: os testes de falha tornaram-se objetivos, validando se o sistema lança a exceção correta no estado proibido.

Finalmente, a **navegabilidade bidirecional** exigiu cautela. Ao encapsular o vínculo na Cliente.AdicionarOrdem e usar um método de fábrica (AbrirOS), garantimos o **sincronismo** dos dois lados (os.Cliente e cliente.Ordens) em um ponto único, evitando estados inválidos e inconsistentes, conforme exigido pelos testes Bidirecional.