Atividade Controller e GRASP

Nome: Francisco Kleber de Souza Oliveira

1. Padrão Controller: Qual foi a classe que implementou o padrão Controller na nossa atividade? Qual é o papel principal dessa classe, e quem ela representa no sistema (em termos de ponto de entrada)?

-A classe seria “ControladorVendas” e seu papel é atuar como um intermediário, recebendo requisições da interface (simulada pela Main) e orquestrando o fluxo de trabalho (como iniciarVenda, adicionarItem e finalizarVenda). Ele delega a lógica de negócio real para os objetos de domínio (como a Venda). Ela representa o ponto de entrada para as requisições do sistema, coordenando um cenário de uso ou subsistema.

2. Controller e Interface: Como o padrão Controller ajuda a desacoplar (diminuir a dependência) a lógica de negócio (Venda, ItemVenda) de uma interface de usuário ou de uma API?

-O Controller age como uma camada de isolamento. A interface (Main) só conhece e chama o ControladorVendas. Ela não precisa saber como a lógica de negócio funciona internamente (como um ItemVenda é criado ou como o total é calculado). O Controller recebe a solicitação e delega para as classes de negócio. Isso evita que a lógica de negócio (como a classe Venda) fique "acoplada" ou misturada com a lógica da interface.

3. Alta Coesão: Qual é a regra fundamental do padrão Alta Coesão? Como a classe Venda demonstrou alta coesão em nossa atividade?

-Atribuir responsabilidades de forma que uma classe seja focada, gerenciável e responsável por um conjunto pequeno e relacionado de tarefas. Deve-se evitar classes que fazem muitas coisas não relacionadas. A classe Venda demonstrou Alta Coesão por focar estritamente em suas responsabilidades de domínio: agregar itens e calcular o total. Ela evitou ter responsabilidades não relacionadas, como lógica de interface de usuário (UI), orquestração de fluxo de trabalho ou persistência em banco de dados.

4. Baixa Coesão (Exemplo): Dê um exemplo de uma responsabilidade que, se fosse atribuída à classe Venda, faria com que ela perdesse sua Alta Coesão (ou seja, teria Baixa Coesão).

-Exemplos mencionados no texto são adicionar métodos como salvarNoBanco() ou enviarEmailAoCliente() à classe Venda. Essas são responsabilidades de infraestrutura ou persistência, não responsabilidades de negócio centrais da Venda, e fariam a classe ter Baixa Coesão por "fazer coisas demais".

5. Controller e Delegação: Por que o ControladorVendas não executa diretamente o cálculo do total da venda? Que padrão GRASP ele utiliza ao "pedir" para a Venda fazer o cálculo?

-O ControladorVendas não executa o cálculo porque sua responsabilidade é orquestrar o fluxo, não executar a lógica de negócio em si. A responsabilidade de calcular o total pertence à Venda, que possui as informações necessárias (a lista de itens). Ao delegar o cálculo para a Venda, ele está respeitando o padrão (Information) **Expert**, pois a Venda é a classe "especialista" que detém os dados para realizar essa operação.

6. Simplicidade da Interface: No nosso método main, que simulava a interface de usuário, por que a interface não precisou saber dos detalhes internos (como criar um ItemVenda ou fazer o cálculo em loop)?

-A interface (Main) interage apenas com o ControladorVendas. O Controller atua como uma fachada ou intermediário que esconde a complexidade interna. A Main simplesmente solicita ações (ex: "adicionar item"), e o Controller cuida de coordenar os objetos de negócio necessários para realizar essa tarefa.

7. Sequência de Eventos: Descreva a sequência de colaboração entre as classes quando o usuário executa a ação de adicionarItem. Quem chama quem?

-A interface (Main) chama o método adicionarItem() no ControladorVendas. O ControladorVendas recebe a chamada e, por sua vez, delega a responsabilidade chamando o método criarItemVenda() no objeto Venda. O objeto Venda (que atua como "Creator") cria a instância de ItemVenda e a adiciona à sua lista interna de itens.

8. Vantagem do Controller: Imagine que a maneira de calcular o total da venda mude no futuro (ex: adiciona-se um imposto). Onde essa mudança deve ser implementada (na classe ControladorVendas ou na classe Venda)? Por quê?

-A mudança deve ser implementada na classe Venda, dentro do método calcularTotal(). Devido aos princípios de Alta Coesão e Information Expert. A classe Venda é a especialista responsável pela lógica de negócio de como seu total é calculado. O ControladorVendas apenas delega a chamada; ele não deve saber *como* o cálculo é feito. Manter a lógica de cálculo na Venda garante que ela permaneça focada e coesa.