ATIVIDADE - PADRÕES COMPOSITE E DECORATOR

1. Siga as instruções e o formato da atividade anterior (implementação Façade e Strategy).

- Busque implementar estes padrões no seu projeto, caso se adeque.

- Caso não se adeque, explique, segundo os fundamentos do padrão, por que o mesmo não pode ser aplicado ao seu projeto.

2. A atividade valerá 75 pontos, sendo 40 pontos do Composite e 35 pontos do Decorator.

3. À esta atividade será somada a pontuação da atividade sobre Decorator, realizada no dia 09/12.

Padrão COMPOSITE

1. **Por que não se aplica no projeto?**

Porque ele objetiva criar hierarquias de objetos com características semelhantes. Estes possuem uma uma interface que é considerada um de seus componentes. E pode ser pensado como um lista de dados especializada no qual todos elementos apresentam a implementação necessária para fazer alguma coisa determinada por um tipo (uma interface). Mas não é só uma lista, pode-se pensar também como uma árvore de objetos, a qual possui métodos que possibilitam realizar ações sem precisar chamar os elementos da hierarquia de forma direta, existindo também uma maneira de adicionar ou retirar objetos nesta hierarquia como em qualquer coleção de dados.

Neste sentido, esse tipo de padrão deve ser utilizado quando se quer representar hierarquias partes-todo de objetos, ou quando é desejado que os clientes sejam capazes de ignorar a diferença entre composições de objetos e objetos individuais. Assim, os clientes tratarão todos os objetos na estrutura composta de maneira uniforme.

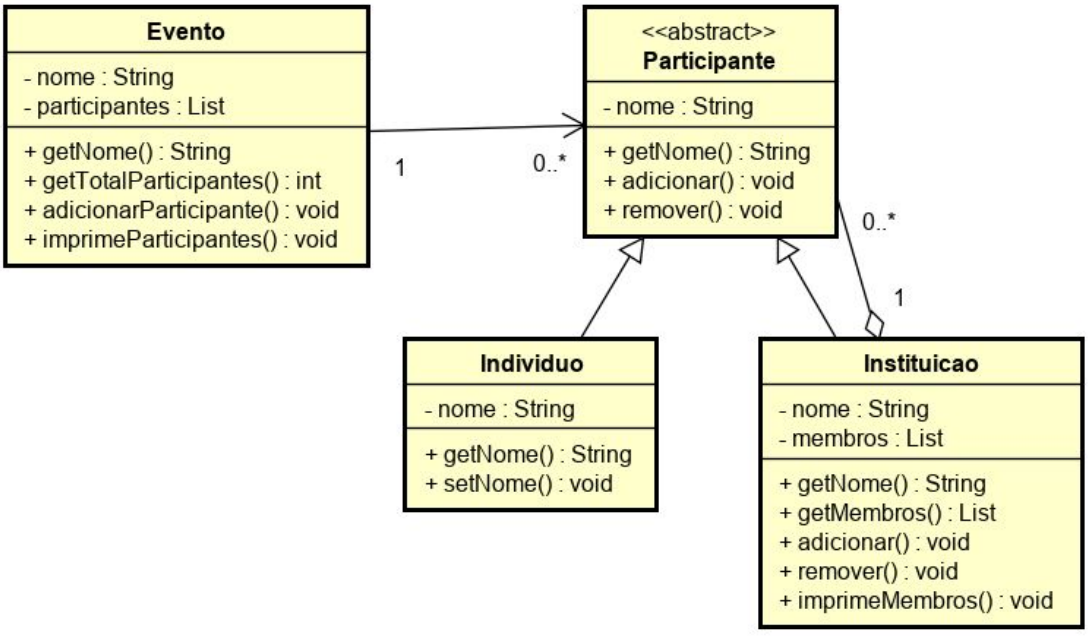
Diante desse contexto, não foi identificado no nosso projeto nenhuma situação a qual pudéssemos aplicar esse tipo de padrão.

2. **Que problema que o padrão está resolvendo neste projeto?**

Como visto anteriormente o Composite é um padrão que utiliza estruturas de árvore para representar hierarquias parte-todo, possibilitando que tanto objetos individuais quanto composições de objetos sejam tratadas pelo cliente uniformemente. Ou seja, ignorando a diferença entre composição de objetos e objetos individuais.

Um exemplo de problema solucionado pelo padrão Composite é o seguinte: Uma instituição de ensino realiza eventos e em tais eventos são inscritos participantes que podem ser um indivíduo ou uma instituição, e cada instituição por sua vez é composta por participantes.

Abaixo é apresentado o diagrama de classe que demonstra a aplicação do padrão como solução. Onde é criada uma classe base (Participante) que possui toda a interface necessária para todos os elementos (Indivíduo e instituição) e é criada também um elemento especial (institução) que agrega outros elementos (sendo eles participantes) e o cliente (evento) que manipula objetos na composição através da interface do componente (participante).



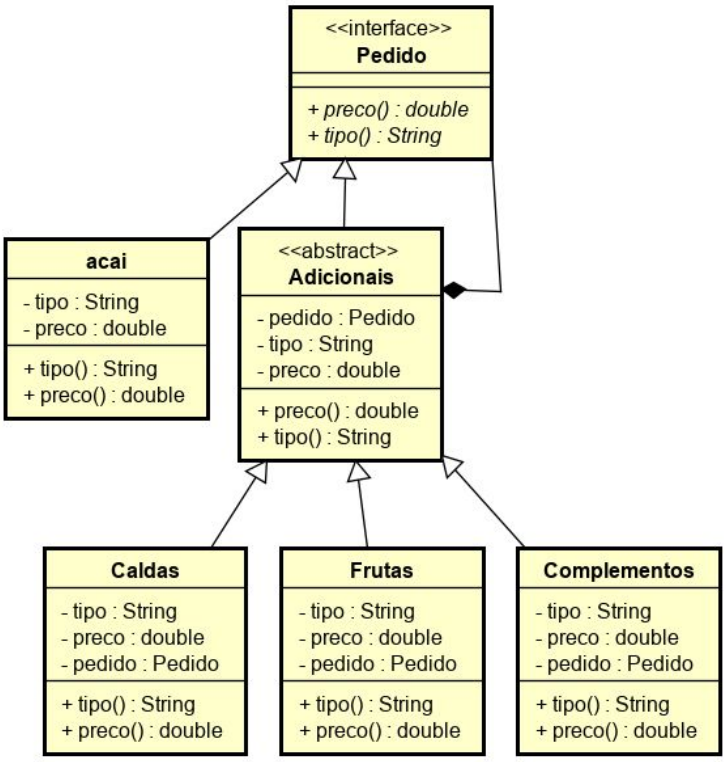
Padrão DECORATOR

1. Por que não se aplica no projeto?

Porque esse tipo de padrão objetiva agregar dinamicamente responsabilidades adicionais a um objeto. Os Decorators fornecem uma alternativa flexível ao uso de subclasses para extensão de funcionalidades e deve ser utilizado quando é desejado acrescentar responsabilidades a objetos individuais de forma dinâmica e transparente, ou seja, sem afetar outros objetos, quando se há responsabilidades que podem ser removidas, ou ainda quando a extensão através do uso de subclasses não é prática. Às vezes, um grande número de extensões independentes é possível e isso poderia produzir uma explosão de subclasses para suportar cada combinação. Ou a definição de uma classe pode estar oculta ou não estar disponível para a utilização de subclasses e no nosso projeto não foi identificado nenhuma dessas situações.

2. Que problema que o padrão está resolvendo neste projeto?

O padrão Decorator será aplicado em um sistema para vendas de Açaí objetivando simplificar o sistema. Cada açaí vendido poderá ser decorado com frutas, complementos e caldas, abaixo será exemplificado por meio do diagrama de classes a aplicação do padrão Decorator nesse sistema .



Na solução apresentada acima, é utilizada uma uma classe abstrata (adicionais) para que as classes concretas (caldas, frutas e complementos) não precisem implementar todos os métodos de negócio de uma interface (pedido). O decorador (adicionais) cria um modelo ao qual outros decoradores podem estender. Portanto, como já existe um Decorator abstrato podem ser adicionados outros Decorators concretos com comportamentos específicos.