Explicação das Implementações

Vinícius Ventura Andreossi, RA:195125
Instituto de Computação - IC Unicamp
(Data: 30 de abril de 2025)

I. IMPLEMENTAÇÃO DA CLASSE MENSAGEM

Foi criada uma classe `Mensagem` que implementa a interface `Notificavel` e representa qualquer forma de comunicação que possa ser visualizada e que produz uma notificação ao receber uma mensagem nova. Foram implementadas as classes `Email` e `SMS` que herdam de `Mensagem`, mas outras formas de comunicação como aplicativos de mensagens de texto, Discord e até mesmo uma abstração de carta também poderiam ser implementadas de maneira semelhante.

A interface `Notificavel` estabelece um contrato em que objetos que a implementem devem possuir dois métodos *void*: `notificar()` e `visualizar()`. O método `notificar()` deve preferencialmente ser executado quando uma mensagem nova é recebida e sua função é gerar alguma forma de aviso para o usuário e.g. uma escrita no terminal ou um som. Já o `visualizar()` deve ser chamado para que o usuário possa ver a mensagem recebida. No meu código eu optei por não permitir uma segunda visualização apenas para que deixar mais evidente que a mensagem já foi visualizada. No entanto, outras implementações podem retirar essa restrição visto que ela não faz parte do contrato estabelecido.

A classe `Mensagem` implementa a interface `Notificável` com o auxílio de uma variável *boolean* `isRead` cuja função é criar um estado para a mesagem e assim monitorar se ela já foi visualizada ou não, de maneira semelhante aos aplicativos de celular. Nas classes filhas o estado é alterado através do método `visualizar()`, mas essa implementação não é feita na classe parente por se tratar de uma classe abstrata.

O diagrama a seguir demonstra a lógica utilizada:

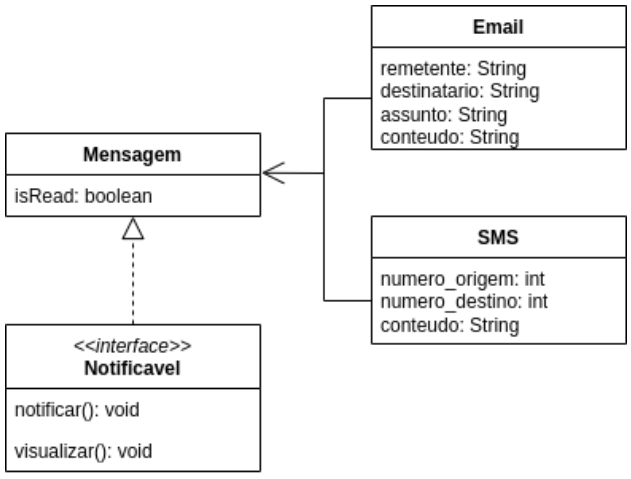

Figura 1: Diagrama da estrutura de classe

II. IMPLEMENTAÇÃO DA HIERARQUIA DE CLASSES DE EVENTOS

A estrutura das classes de eventos para adicionar a função de características de evento foi feita a partir de uma simples mudança na classe base `Evento`. Essa ideia foi baseada no conceito de herança pois, dessa forma, as classes filhas já teriam automaticamente essa funcionalidade. Essa implementação é ilustrada no diagrama a seguir(obs: por simplicidade, apenas os aspectos relevantes para a reimplementação foram adicionados ao diagrama):

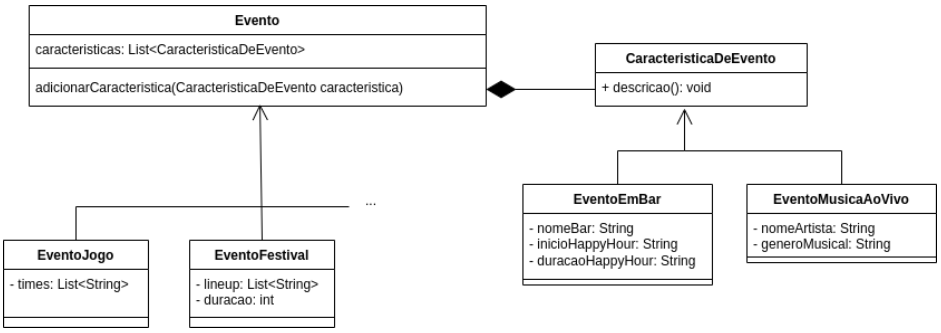

Figura 2: Diagrama da hierarquia da classe de eventos

A ideia foi inserir as classes filhas de `CaracteristicaDeEvento` por composição na classe base através de uma lista de objetos `CaracteristicaDeEvento`. Essa implementação permite flexbilidade no número de características que cada evento pode ter, visto que alguns eventos podem ter várias características e outros podem não ter nenhuma. As características são atrbuídas através do método `adicionarCaracteristica(...)`.