Padrão Chain of Responsability

Padrão Comportamental

I. Problema: Filtro de Acesso em um servidor

O padrão **Chain of Responsibility** não é um padrão no qual encontramos aplicabilidade frequente em programas simples, haja vista que sua relevância é oportuna em situações na qual o código opera com cadeias de objetos.

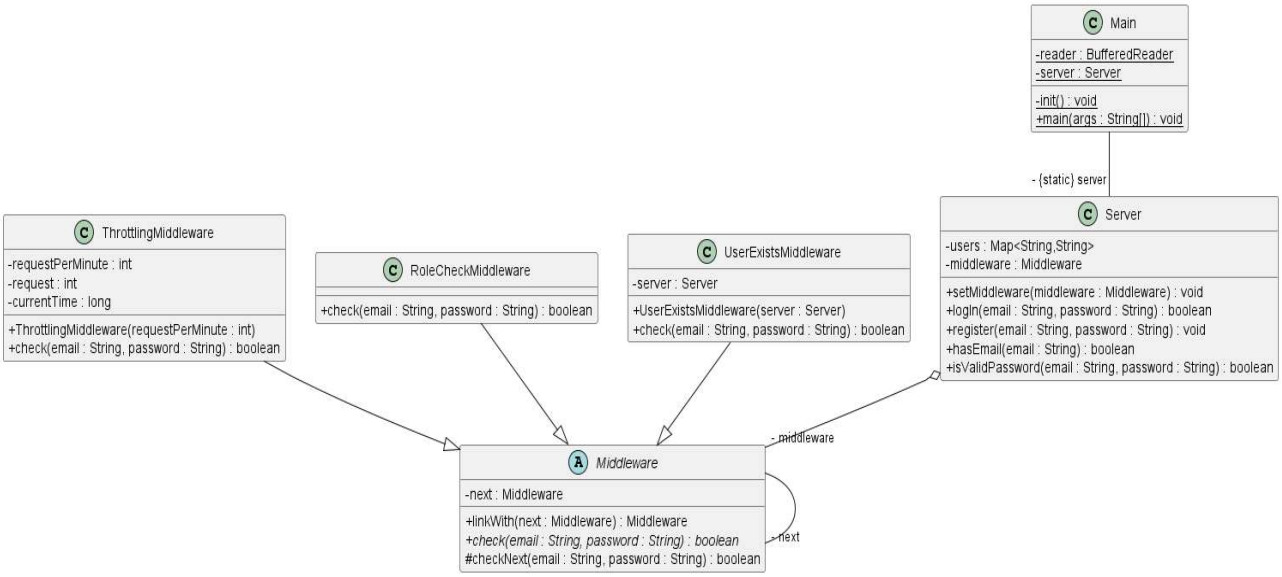
Um dos casos de uso mais populares para o padrão são os **filtros de acesso seqüencial**. O filtro é usado para interceptar a solicitação do cliente e fazer algum pré-processamento. Por exemplo, em uma aplicação web cliente-servidor, temos a situação em que uma solicitação/requisição que contém dados do usuário passa por uma cadeia sequencial de manipuladores que executam várias tarefas, como autenticação, autorização e validação.

No problema em questão, o usuário digitará o login e senha para autenticação em um servidor web. A cadeia de responsabilidades passa pelas seguintes componentes:

```
ThrottlingMiddleware > UserExistsMiddleware > RoleCheckMiddleware
        (1)                    (2)                    (3)
```

(1) **ThrottlingMiddleware**: Verifica se a requisição não excede o limite de requisições por minuto
(2) **UserExistsMiddleware:** Verifica se as credenciais de acesso do usuário são válidas (login e senha)
(3) **RoleCheckMiddleware:** Identifica o nível de acesso do usuário (administrador ou usuário)

II. Diagrama de Classe



III. A tarefa

Nesta tarefa, veremos como aplicamos o padrão **chain of responsibility** em um exemplo de **filtro de acesso**.

Este exemplo é um pouco diferente da versão canônica do padrão fornecida por vários autores. A maioria dos exemplos do padrão é construída com a noção de procurar o *handler* correto, iniciá-lo, e sair da cadeia depois disso. Mas aqui, executamos todos os *handlers* até que aquele que **não possa lidar** com uma solicitação seja encontrado. Esteja ciente de que esse ainda é o padrão **Chain of Responsibility**, mesmo que o fluxo seja um pouco diferente.