Padrão observer

Certified Developer
The Ultimate Tech Degree

DigitalHouse >
Coding School

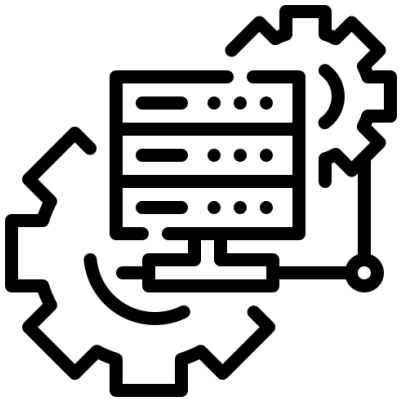
Índice

1 Contextualização

Propósito

Um determinado objeto pode ter outros **dependentes** dele. Esses outros objetos podem precisar ser **atualizados** com base em uma mudança de estado no **objeto** do qual eles dependem. Ao tentar implementar essa lógica, muitas dificuldades surgem.

O padrão **Observer** propõe uma solução criando uma interface que, quando um objeto muda de estado, todos os objetos que dependem dele são notificados e atualizados **automaticamente**.



Solucão

É criada uma interface para o ObjectObserver (Observer) e uma outra para os Observadores (Observador). A classe concreta a ser observada implementa a interface Observable e os observadores concretos implementam Observer. Essas duas interfaces possuem um método que as classes concretas devem declarar, e por meio desses métodos é que os observadores serão atualizados a cada mudança de estado no ObjectObserver.

Assim, as atualizações de status sempre serão obtidas independentemente do tipo de objeto que sejam os observadores e o sujeito observável.

Vantagens e desvantagens

✓	Permite modificar os sujeitos e observadores de forma independente. É possível reutilizar objetos sem reutilizar seus observadores ou vice-versa. Isso adiciona escalabilidade, permitindo que observadores sejam adicionados sem modificar o assunto ou outros observadores.
✓	Ao contrário de uma solicitação comum, a notificação enviada por um assunto não precisa especificar seu destinatário. Isso gera um alcance a todos os observadores interessados.
✓	Graças ao fato de que sujeito e observador não estão fortemente acoplados, eles podem pertencer a diferentes camadas de abstração de um sistema.
✗	Uma atualização aparentemente inofensiva sobre o assunto pode gerar uma série de atualizações em cascata dos observadores e seus objetos dependentes. Isso pode levar a atualizações falsas que são muito difíceis de localizar.

2 Diagrama UML

Diagrama Padrão *Observer*

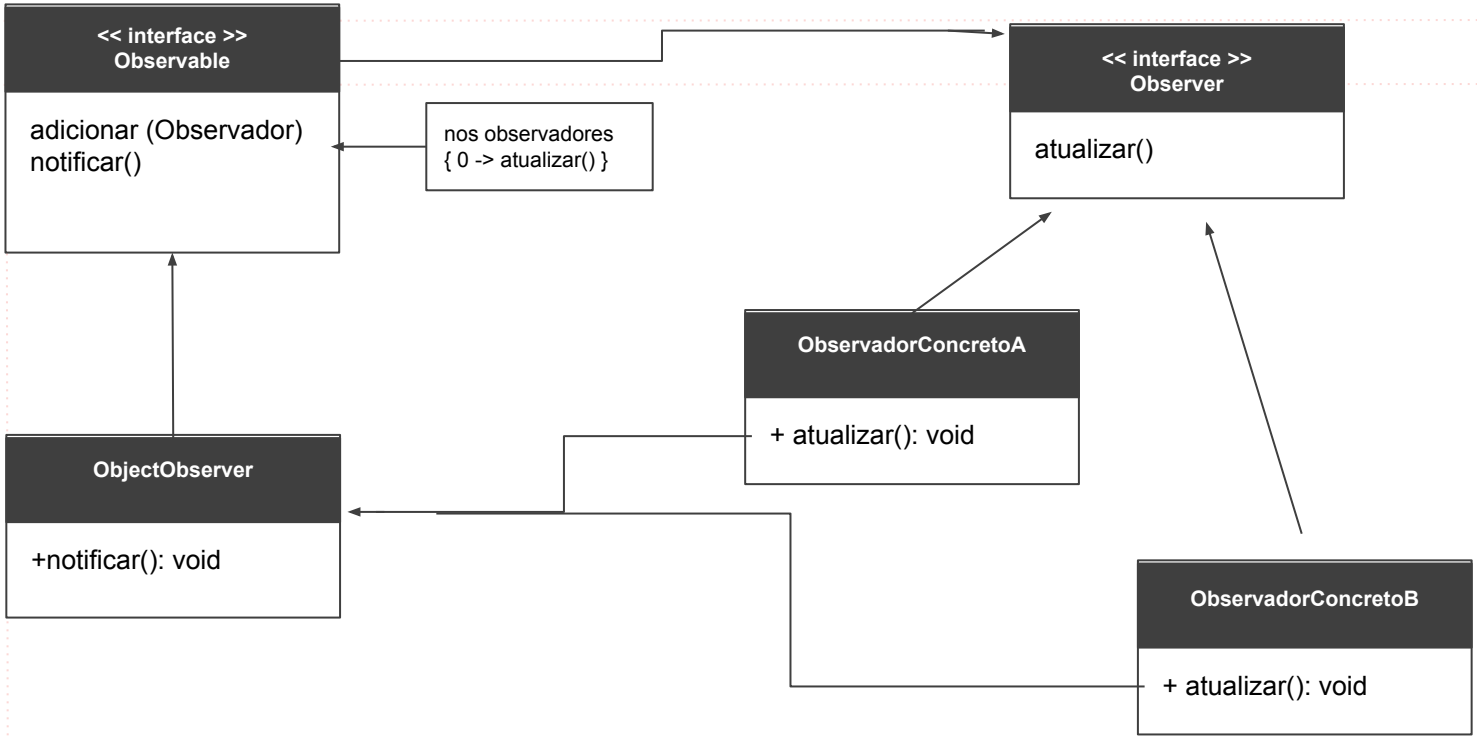
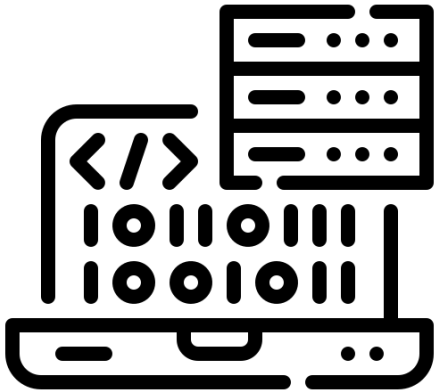


Diagrama Padrão *Observer*

- Interface observable (assunto): cada implementação (assunto) conhece seus observadores e pode ser observada por qualquer número de observadores. Ele também fornece uma interface para adicionar ou remover observadores.
- Observer: define uma interface para que cada implementação (Concrete Observer) possa atualizar os objetos que devem ser notificados de mudanças no assunto.
- SpecificSubject: Envie uma notificação aos seus observadores quando seu status mudar.
- Observador concreto: mantém uma referência a um objeto Assunto concreto. Salva um estado que deve ser consistente com o do Assunto. Implemente a interface de atualização do Observer para manter seu status em sincronia com o do assunto.

Como funciona?

Quando o SubjectObservable passa por uma mudança de estado, o método **"notificar ()"** é executado, o qual percorre uma lista contendo todos os objetos que observam o SubjectObservable e chama seu método **"atualizar ()"**. Desta forma, todos os observadores ficam atualizados em caso de qualquer alteração, sem a necessidade de verificação constante de atualizações de status.

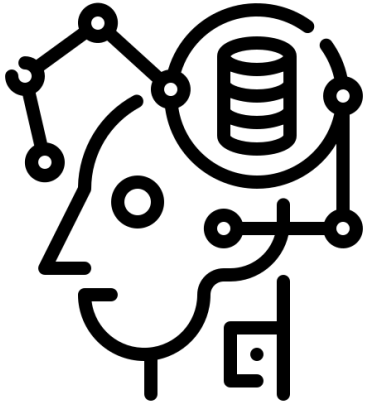


Conclusão

O padrão cria uma dependência direta de cada observador em relação ao assunto. Embora isso possa levar a complicações, é a maneira como o padrão é estruturado.

O que não deveria acontecer é que o sujeito depende de um observador. Este comportamento violaria a ignorância que o sujeito deve ter sobre seus observadores e poderia gerar dependências cíclicas.

É conveniente especificar as modificações de interesse explicitamente. A eficiência pode ser melhorada estendendo a interface de registro de assunto para permitir que os observadores registrem apenas os eventos específicos que os interessam.




DigitalHouse>
Coding School