Padrão de Projeto State

O padrão de projeto State pertence ao grupo dos padrões comportamentais, ele lida com a maneira como os objetos interagem e se comportam. Esse padrão é útil quando um objeto precisa mudar seu comportamento dependendo do seu estado interno, mas queremos evitar o uso de muitos condicionais (como if ou switch) que deixam o código confuso e difícil de manter.

Problema que o Padrão State Resolve

Imagine que você está criando um jogo de luta, e cada lutador pode estar em um dos seguintes estados:

- Atacando
- Defendendo
- Descansando

Dependendo do estado atual do lutador, ele deve agir de forma diferente. Por exemplo:

- Quando ele está atacando, pode infligir dano no oponente.
- Quando está defendendo, pode reduzir o dano recebido.
- Quando está descansando, ele pode recuperar energia.

Agora, se você quiser implementar esses comportamentos sem usar o padrão State, você provavelmente acabaria usando vários if ou switch espalhados pelo código, para verificar o estado do lutador e determinar a ação correta. Isso faz o código crescer rapidamente e se tornar difícil de entender e manter, especialmente à medida que novos estados são adicionados.

Um exemplo de como podemos implementar essa classe:
https://gist.github.com/paeeglee/cb8cefec0d68f6e86a42004247874c1e

Como o mesmo contexto ficaria quando utilizamos o padrão State.

Passo 1: Criando a Interface de Estado

Criamos uma interface que define o que cada estado deve fazer. Vamos definir o método executarAcao() para que cada estado tenha seu próprio comportamento.

https://gist.github.com/paeeglee/4e26556f7bbcc8e97c16a29b638e0c7f

Passo 2: Criando os Estados Concretos

Criamos classes para cada estado concreto: Atacando, Defendendo, e Descansando. Cada uma delas implementa a interface Estado.

https://gist.github.com/paeeglee/357caf95102a630c115b3b7065c055f3

Passo 3: Criando o Contexto

Criamos a classe Lutador, que é o contexto. Ela contém uma referência para o estado atual, que pode ser alterado conforme necessário. O método `executarAcao()` vai delegar o comportamento para o estado atual.

https://gist.github.com/paeeglee/d5e88567d869efb4e2df19a7946c074d

Passo 4: Utilizando o Código

Agora que o padrão State foi implementado, podemos criar um lutador e alterar seu comportamento dinamicamente, mudando seu estado.

https://gist.github.com/paeeglee/eddab4497b9e74aa4c8cc760c0c01ad8

Vantagens do Padrão State

- Organização: Separar o comportamento de cada estado em classes distintas torna o código mais organizado e fácil de manter.
- Extensibilidade: É fácil adicionar novos estados sem modificar as classes existentes, respeitando o princípio do Open/Closed do SOLID.
- Remoção de Condicionais: O uso de muitos `if-else` ou `switch` é eliminado, tornando o código mais limpo.

Desvantagens do Padrão State

- Aumento no número de classes: Cada novo estado requer a criação de uma nova classe, o que pode resultar em muitos arquivos de código, dependendo da complexidade do sistema.
- Complexidade Adicional: Em sistemas simples, o uso do padrão State pode ser excessivo. Ele se torna mais útil à medida que a lógica de estados fica mais complexa.

Quando Usar o Padrão State?

- Quando um objeto precisa mudar seu comportamento baseado em seu estado atual.
- Quando você quer evitar o uso de múltiplos condicionais `if-else` ou `switch` para lidar com diferentes comportamentos de um objeto.
- Quando o comportamento associado a um estado é complexo ou tende a crescer com o tempo.