State

Padrões de Projeto
Dra. Alana Morais

Sumário

O que já foi visto?

- Padrões Comportamentais: Template Method e Observer

Hoje:

- State

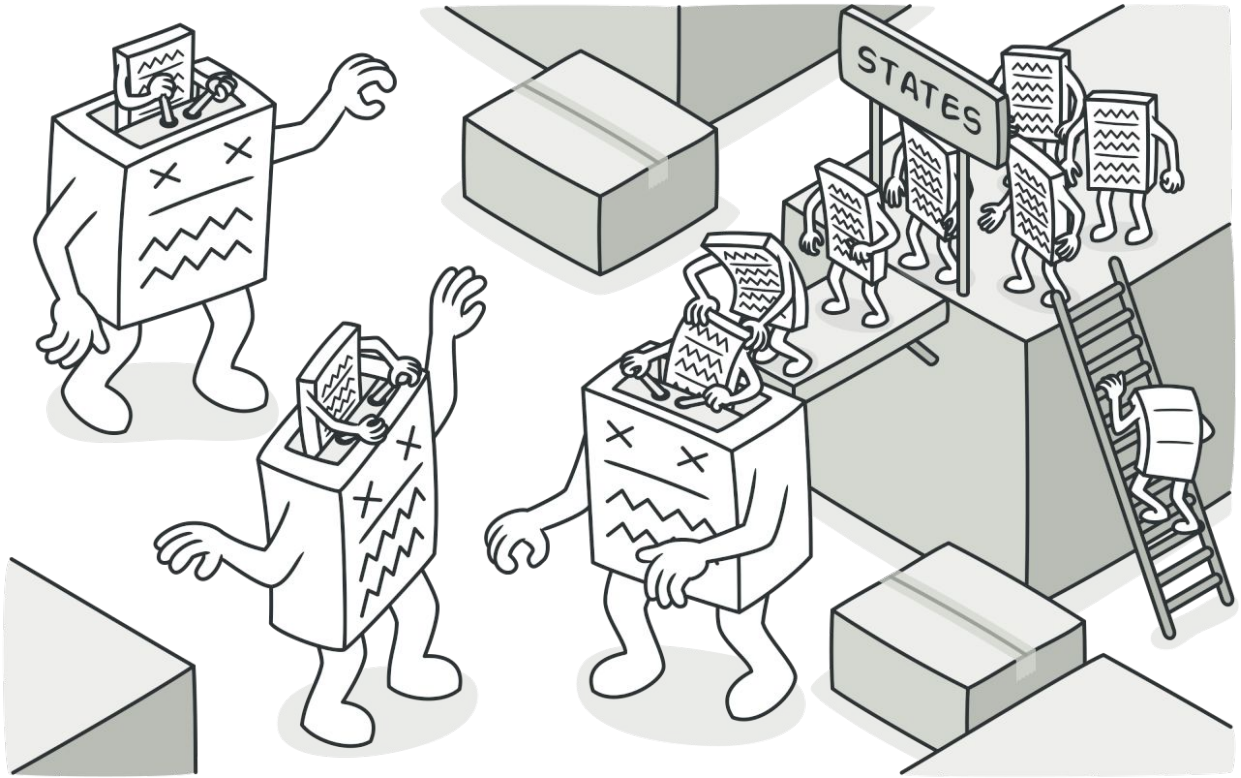
State

PERMITIR QUE UM OBJETO ALTERE SEU COMPORTAMENTO QUANDO SEU ESTADO INTERNO É ALTERADO. PARECE QUE O OBJETO MUDOU SUA CLASSE.

State

- O padrão encapsula os estados em classes separadas e delega as tarefas para o objeto que representa o estado atual, nós sabemos que os comportamentos mudam juntamente com o estado interno.
- O padrão State é motivado por aqueles objetos que, em seu estado atual, varia o seu comportamento devido as diferentes mensagens que possa receber.
- Também conhecido como:
    - Objects for States

State - Problema



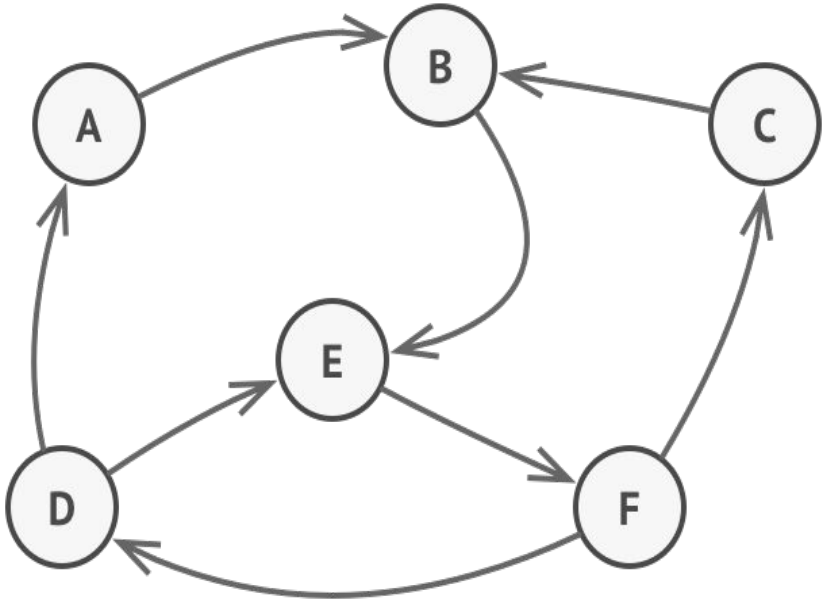
State

Muito popular em contextos que envolvem Máquinas de Estados

Onde já ouvimos falar sobre Estados?

Outras Disciplinas?

State - Máquina de Estados



A idéia principal é que, em qualquer momento, há um número finito de estados em que um programa pode estar.

Dentro de qualquer estado único, o programa se comporta de maneira diferente, e o programa pode ser alternado de um estado para outro instantaneamente.

Dependendo do estado atual, o programa pode ou não alternar para outros estados.

Essas regras de comutação, chamadas de transições, também são finitas e predeterminadas.

State - Problema

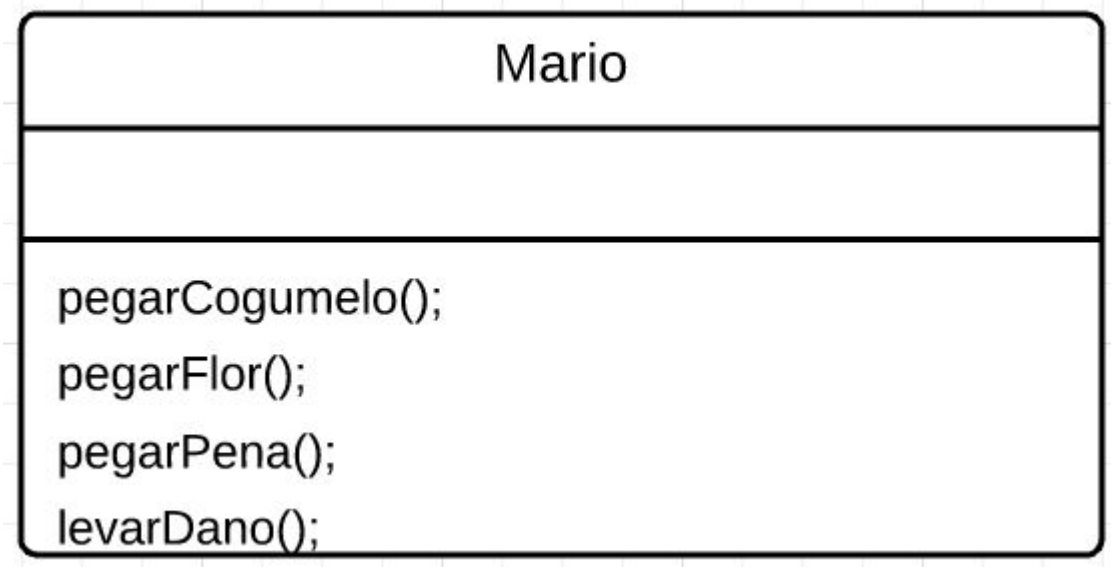
- Durante o jogo Mario acontecem várias trocas de estado com o Mario, por exemplo, ao pegar uma flor de fogo o mario pode crescer, se estiver pequeno, e ficar com a habilidade de soltar bolas de fogo.
- Desenvolvendo um pouco mais o pensamento temos um conjunto grande de possíveis estados, e cada transição depende de qual é o estado atual do personagem. Como falado anteriormente, ao pegar uma flor de fogo podem acontecer quatro ações diferentes, dependendo de qual o estado atual do mario:
    - Se Mario pequeno -> Mario grande e Mario fogo
    - Se Mario grande -> Mario fogo
    - Se Mario fogo -> Mario ganha 1000 pontos
    - Se Mario capa -> Mario fogo
- Todas estas condições devem ser checadas para realizar esta única troca de estado. Agora imagine o vários estados e a complexidade para realizar a troca destes estados: Mario pequeno, Mario grande, Mario flor e Mario penal.

State - Problema

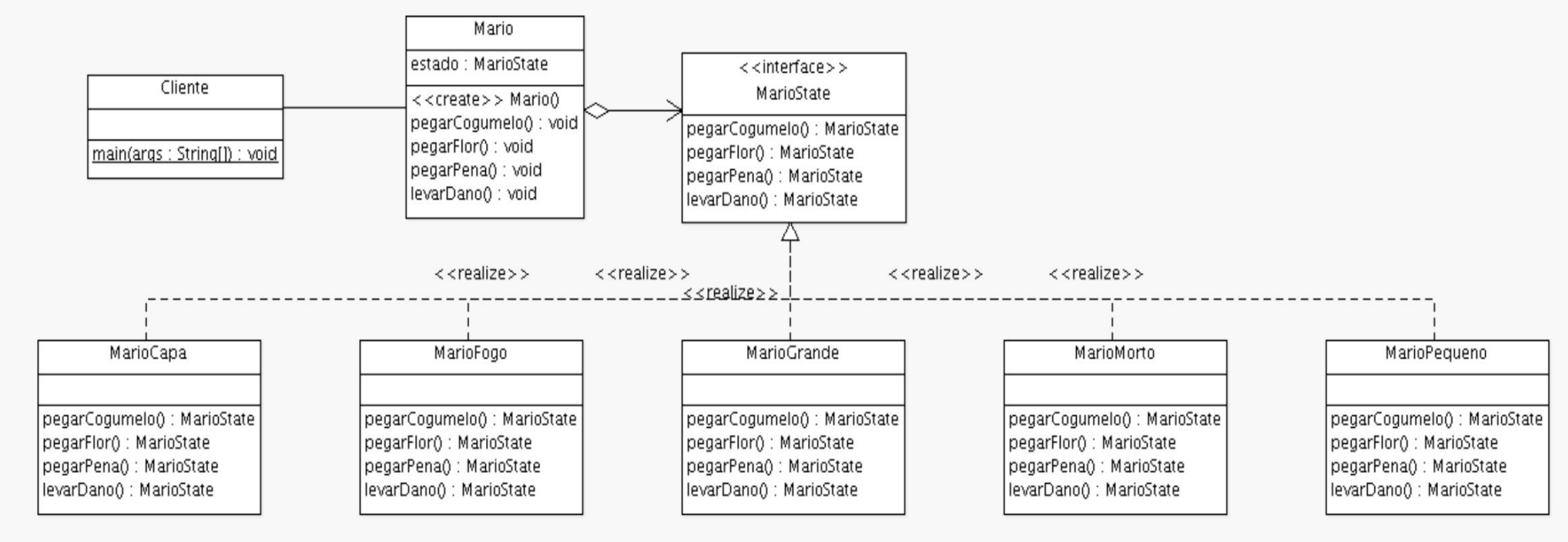
- Pegar Pena:
  - Se Mario pequeno -> Mario grande e Mario capa
  - Se Mario grande -> Mario capa
  - Se Mario fogo -> Mario fogo
  - Se Mario capa -> 1000 pontos
- Levar Dano:
  - Se Mario pequeno -> Mario morto
  - Se Mario grande -> Mario pequeno
  - Se Mario fogo -> Mario grande
  - Se Mario capa -> Mario grande
- Pegar Cogumelo:
  - Se Mario pequeno -> Mario grande
  - Se Mario grande -> 1000 pontos
  - Se Mario fogo -> 1000 pontos
  - Se Mario capa -> 1000 pontos
- Pegar Flor:
  - Se Mario pequeno -> Mario grande e Mario fogo
  - Se Mario grande -> Mario fogo
  - Se Mario fogo -> 1000 pontos
  - Se Mario capa -> Mario fogo

State - Solução

Solução centrada em if-else

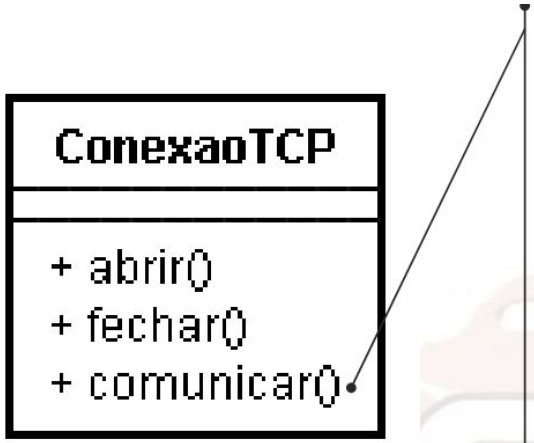


State - Solução



State - Exercício

- Implemente com *State* uma solução que melhore o controle da conexão TCP (three-way handshake)
- Conexão possui um objeto que representa seu estado e implementa o método que depende deste estado.




```
if (conexaoAberta()) {
  // Envia a comunicação.
}
else if (recebendoDados()) {
  // Aguarda terminar a comunicação.
}
else if (conexaoFechada()) {
  // Erro ou reconexão automática.
}
```

State - Quando Usar?

- O comportamento de um objeto depende do seu estado, que é alterado em tempo de execução;
- Operações de um objeto possuem condicionais grandes e com muitas partes (sintoma do caso anterior).

State - Vantagens e Desvantagens

+ Princípio da responsabilidade única. Organize o código relacionado a estados específicos em classes separadas.
+ Princípio Aberto / Fechado. Introduza novos estados sem alterar as classes de estado existentes ou o contexto.
+ Simplifique o código do contexto eliminando condicionais volumosos de máquinas de estado.
- Aplicar o padrão pode ser um exagero se uma máquina de estado tiver apenas alguns estados ou raramente for alterada.

State - Outros Padrões

- É comum que objetos estados obedeçam a outros dois padrões: Singleton e Flyweight.
  - Estados singleton são capazes de manter informações, mesmo com as constantes trocas de estados.
  - Estados Flyweight permitem o compartilhamento entre objetos que vão utilizar a mesma máquina de estado.
- O padrão State tem semelhanças com outros dois padrões: Strategy e Bridge.
  - A ideia é muito parecida com o Strategy: eliminar vários ifs complexos espalhados utilizando subclasses.
  - A diferença básica é que o State é mais dinâmico que o Strategy, pois ocorrem várias trocas de objetos estados, os próprios objetos estados realizam as transições.

Dúvidas?

alanamm.prof@gmail.com

Referências

Material de Vítor E. Silva Souza

Gamma et al. Design Patterns.

https://refactoring.guru/design-patterns