AULA 11 – PADRÃO STATE

GSI020 – Programação Orientada a Objetos II

Prof. Dr. Murillo G. Carneiro
mgcarneiro@ufu.br

UFU

Material baseado nos slides cedidos pelo Prof. Rafael D. Araújo (FACOM/UFU)

Objetivo da aula

- Entender o funcionamento do padrão de projeto *State* (Estado).

Motivação

- Comportamento de máquinas de venda automática
  - *O que acontece quando não possui mais troco disponível? * pede o valor exato!*
- Representação de estados de conexão de uma rede de comunicações

State

- É um padrão de projeto de propósito **comportamental** com escopo de **objetos**.
- Propõe uma solução para tornar o comportamento de um objeto dependente do seu estado interno.

Intenção

- Permitir a um objeto alterar seu comportamento quando seu estado interno muda. O objeto parecerá ter mudado sua classe.

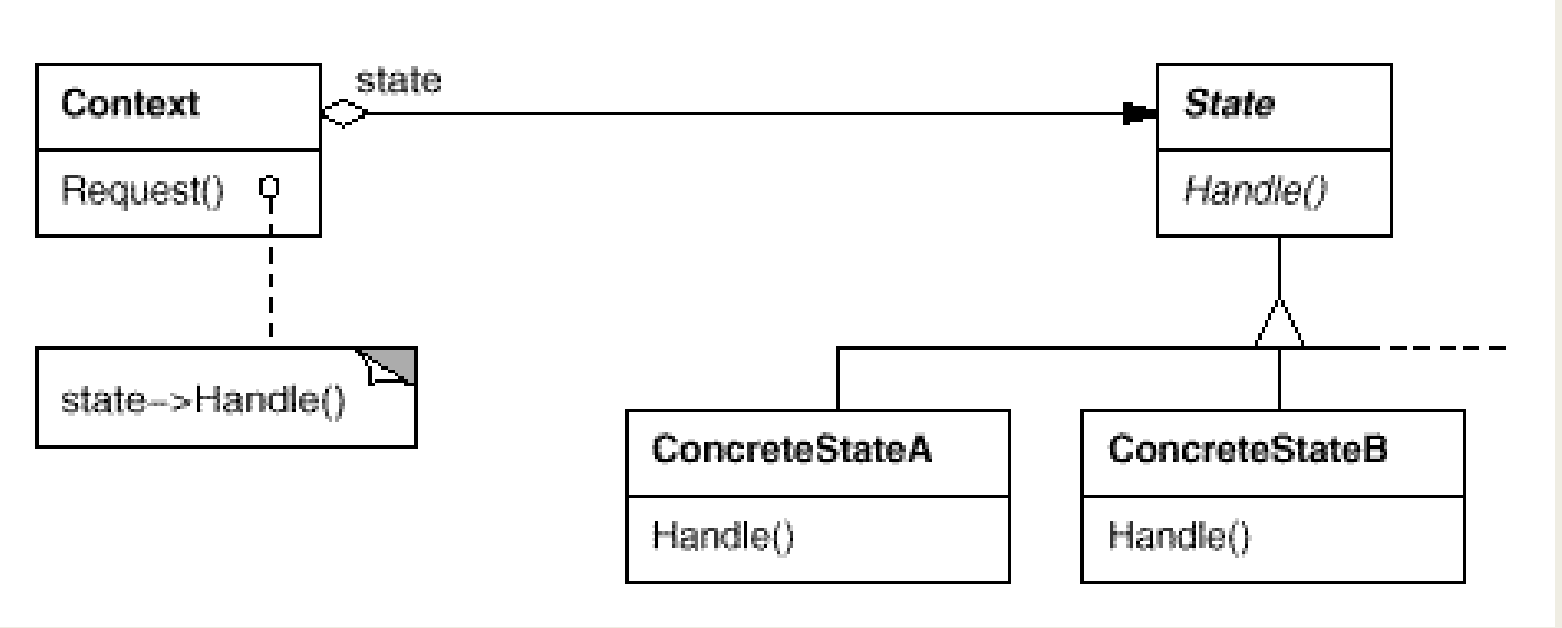
Quando usar

- O comportamento de um objeto depende do seu estado e ele pode mudar seu comportamento em tempo de execução, dependendo desse estado
- Operações têm comandos condicionais grande, com várias alternativas que dependem do estado do objeto

Elementos participantes

- **State**: define uma interface para encapsulamento associado com um determinado estado
- **ConcreteState**: subclasses que implementam um comportamento específico associado com um estado
- **Context**: define a interface de interesse para os clientes e mantém uma instância de uma subclasse ConcreteState que define o estado corrente

Estrutura


Context
Request()
state
State
Handle()
state->Handle()
ConcreteStateA
Handle()
ConcreteStateB
Handle()

Benefícios

Princípio da abertura-fechamento (OCP): classe fechada para alterações de estados, mas os estados são abertos a extensões.

- Consegue lidar com o que o objeto é em um determinado momento
  - *Encapsula o comportamento dependente do estado*
- O contexto não precisa conhecer os estados (if..else)
  - *A lógica de transição de estados é implementada pelos próprios estados*

Desvantagens

- Aumento no número de objetos existentes
  - *divisão da lógica dos estados em diferentes classes*
- Quem define as transições de estado?
  - *Se for a classe Context: ok para situações simples*
  - *Se forem as classes concretas de estados: mais flexível mas causa maior dependência entre elas*
- Quando os estados concretos são criados?
  - *Quando necessário ou criar todos de uma vez só*

Implementação

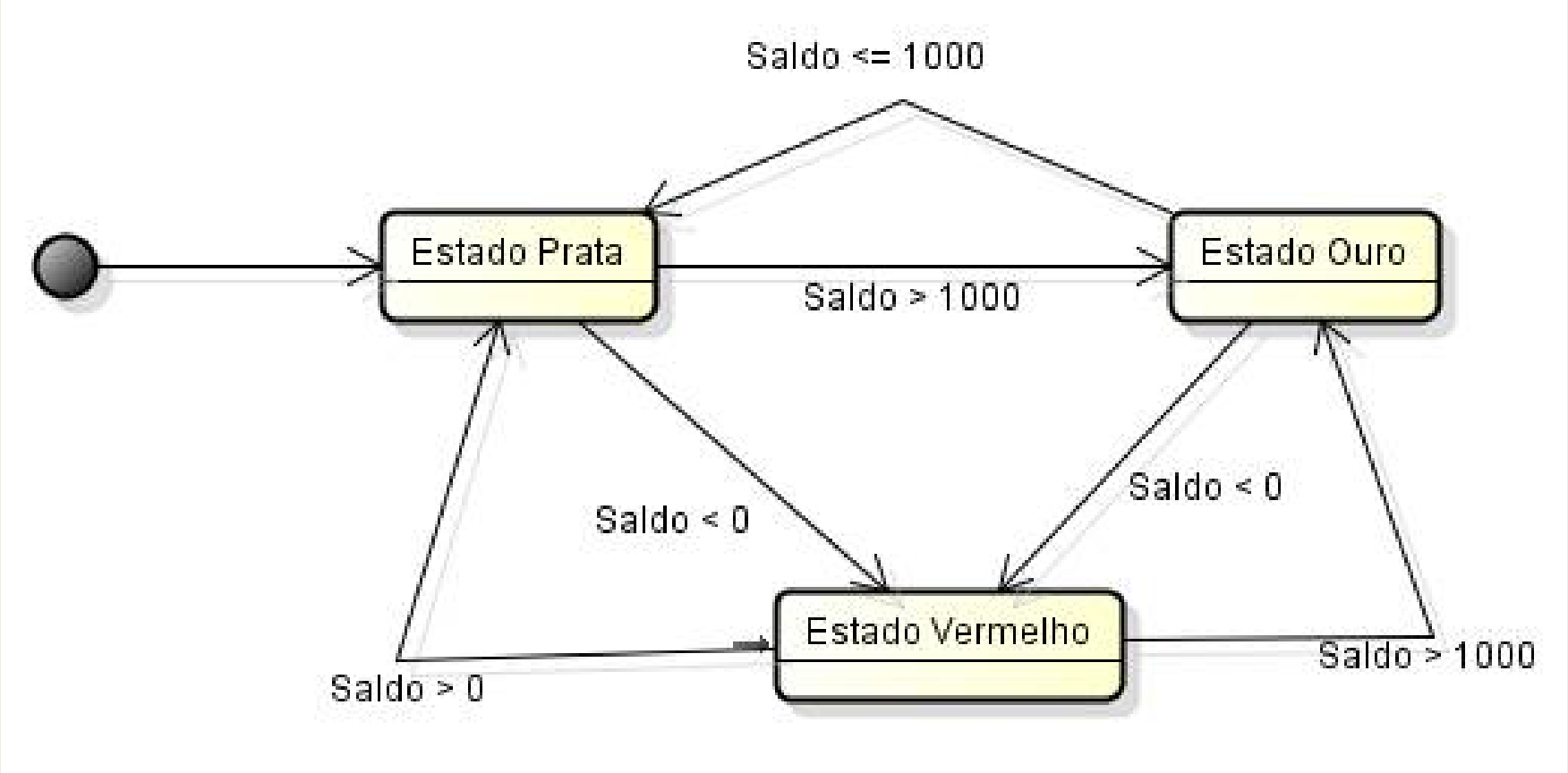
- Especificar os diferentes estados que um objeto pode assumir
  - *Criar uma interface comum*
  - *Criar uma classe para cada estado*
- Identificar as transições de estados
- Criar a classe contexto que possui um estado específico

Problema prático

Controle de estado de contas bancárias:

- Se a conta estiver com saldo entre 0 e 1000 = estado **prata**. Neste caso, o cliente deve pagar uma taxa de R$ 1,00 por cada saque que realizar
- Se a conta estiver com saldo maior que 1000 = estado **ouro**. Neste caso, o cliente não paga taxa de saque e ainda ganha um rendimento imediato (0,1%) em cada depósito que realizar
- Se a conta estiver com saldo negativo = estado **vermelho**. Neste caso, não é permitido a realização de saques. Apenas depósitos

Problema prático



Referências

- GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1995. Capítulo 5.