Projeto de Software

Modelagem de Classes de Projeto

Lesandro Ponciano

2024

Objetivos da Aula

- Revisar conceitos de diagrama de classes
- Discutir os Diagramas de Classes de Projeto
  - Tipos de classes e seus estereótipos
  - Associações versus Dependências
  - Classes parametrizadas
  - Herança de Implementação e de Interface
  - Classificação Dinâmica

Representação de Classes

- De forma geral, as classes são mais detalhadas, exemplo visibilidade:
  - Pública (+), outras classes podem ter acesso ao atributo
  - Privada (-), somente é acessado diretamente pela própria classe
  - Protegida (#), utilizado com restrições, exemplo além dos objetos da subclasse,os métodos de suas subclasses também poderão enxergá-los
  - Pacote (~), que é acessado pelo relacionamento da classe com a classe externa.

Tipos de Classes

- Na análise, o diagrama de classes só contém classes do domínio
- No projeto, há também outros tipos de classe
- Tipos de classes
  - Entidade
  - Fronteira
  - Controle
  - Outras

Classes de Entidade (*Entity*)

- Características
  - Representam algum conceito encontrado no domínio do problema
  - Representam os conceitos do domínio que o sistema deve processar
  - Classes "normais" geralmente obtidas na etapa de engenharia de requisitos do sistema
  - A nível de projeto são mais detalhadas:
    - incluem os tipos de dados para os atributos, parâmetros e retornos
    - Atributos que não existem no domínio, como os chamados “atributos de implementação”, que são códigos a serem usados como identificadores únicos
- Exemplos:
  - Lógica de negócio: Aluno, Professor, Disciplina
  - Persistência: AlunoTemDisciplina

Classes de Fronteira (*Boundary*)

- Características
  - Não são objetos do domínio
  - Devem servir apenas como um ponto de captação/apresentação de informações do/ao ambiente
  - Trata de interfaces externas com usuários, sistemas, hardware
- Exemplos
  - Clientes WEB clássicos: a classe de fronteira é representada por páginas HTML estáticas e/ou dinâmicas
  - Clientes móveis: exemplo: classes de fronteira que implementam algum protocolo específico como ambiente, tal como WML
  - Clientes *stand-alone*: Interface gráfica como as produzidas com Swing/JFC da linguagem Java
  - Serviços WEB: *web services*

Classes de Controle (*Control*)

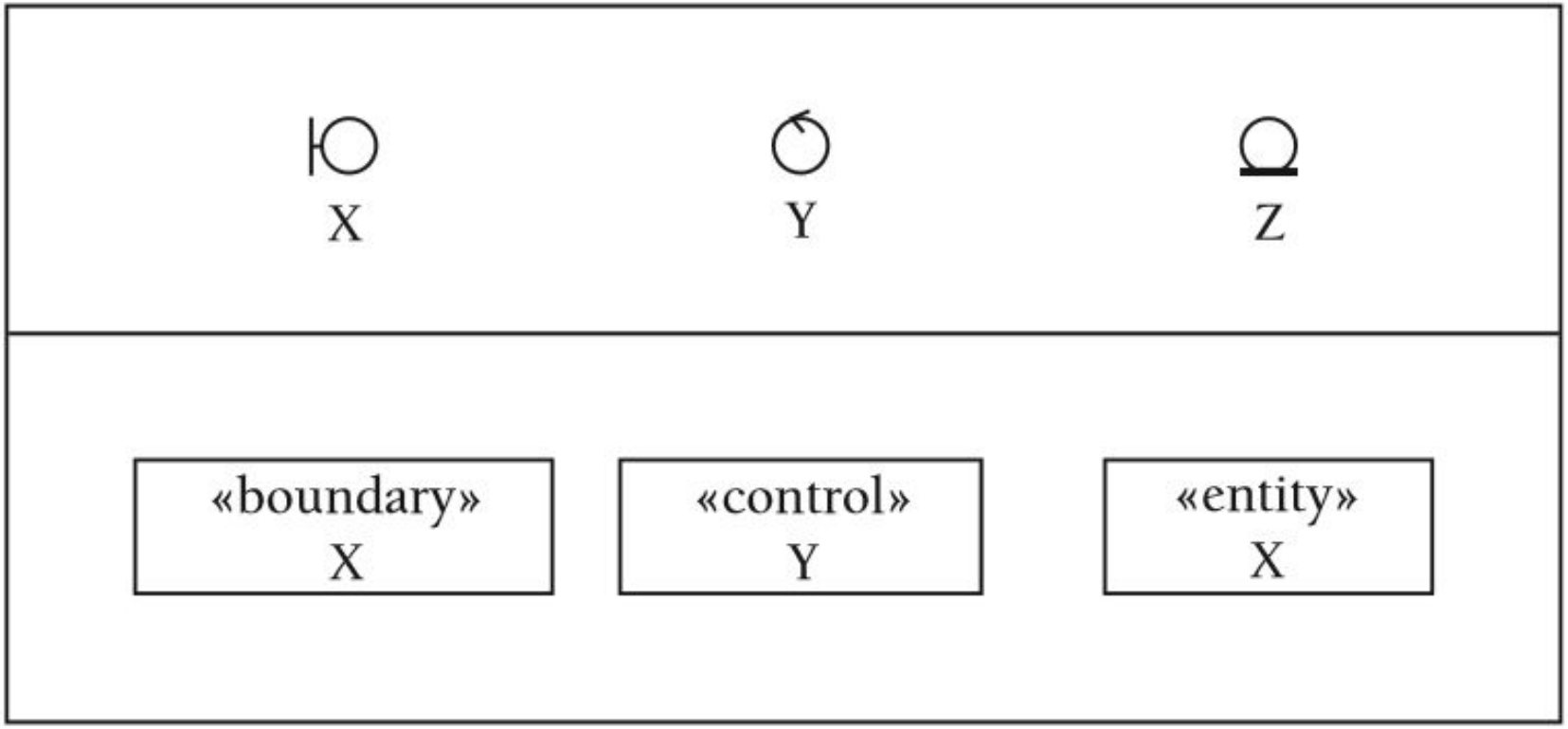
- Também é chamada de controlador
  - Não são objetos do domínio
  - Coordenam a execução de alguma funcionalidade específica
    - Decidem o que o sistema faz quando ocorre um evento externo
  - Coordenam a interação entre outros objetos
  - São pontes entre objetos de fronteira e objetos de entidade
- Consistem em controladores do GRASP

Outras Classes

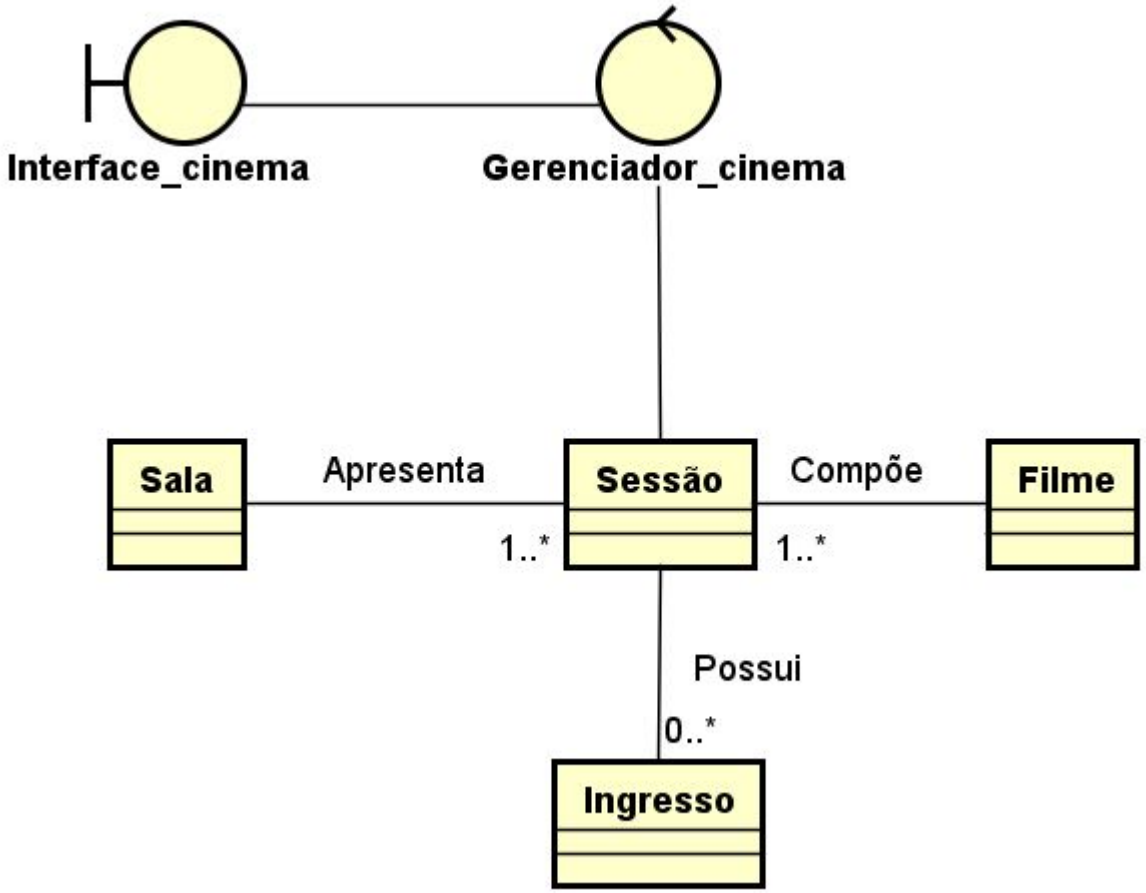
- Classes que nãos são de domínio, fronteira ou controle

- Surgem de
  - Padrões de projeto empregados
  - Bibliotecas empregadas
  - *Frameworks* usados

Esteriótipos para Classes



Exemplo (incompleto)



Representação de Associações

- Na análise
  - Associações (composição, agregação)
  - Herança
- No projeto
  - O conceito de associação se transforma do conceito de dependência
  - Uma dependência entre classes indica que uma classe depende dos serviços fornecidos pela outra

Tipos de Dependências

1) Dependência por atributo
   - *A* possui um atributo cujo tipo é *B*
   - A dependência por atributo é também chamada de dependência estrutural, as demais são dependências não estruturais
2) Dependência por variável global
   - *A* utiliza uma variável global cujo tipo é *B*
3) Dependência por variável local
   - *A* possui alguma operação cuja implementação utiliza uma variável local de tipo *B*
4) Dependência por parâmetro
   - *A* possui pelo menos uma operação e esta possui pelo menos um parâmetro, cujo tipo é B

Notação de Dependência

- Seta tracejada ligando as classes envolvidas
- O sentido da seta é da classe dependente para a classe da qual ela depende

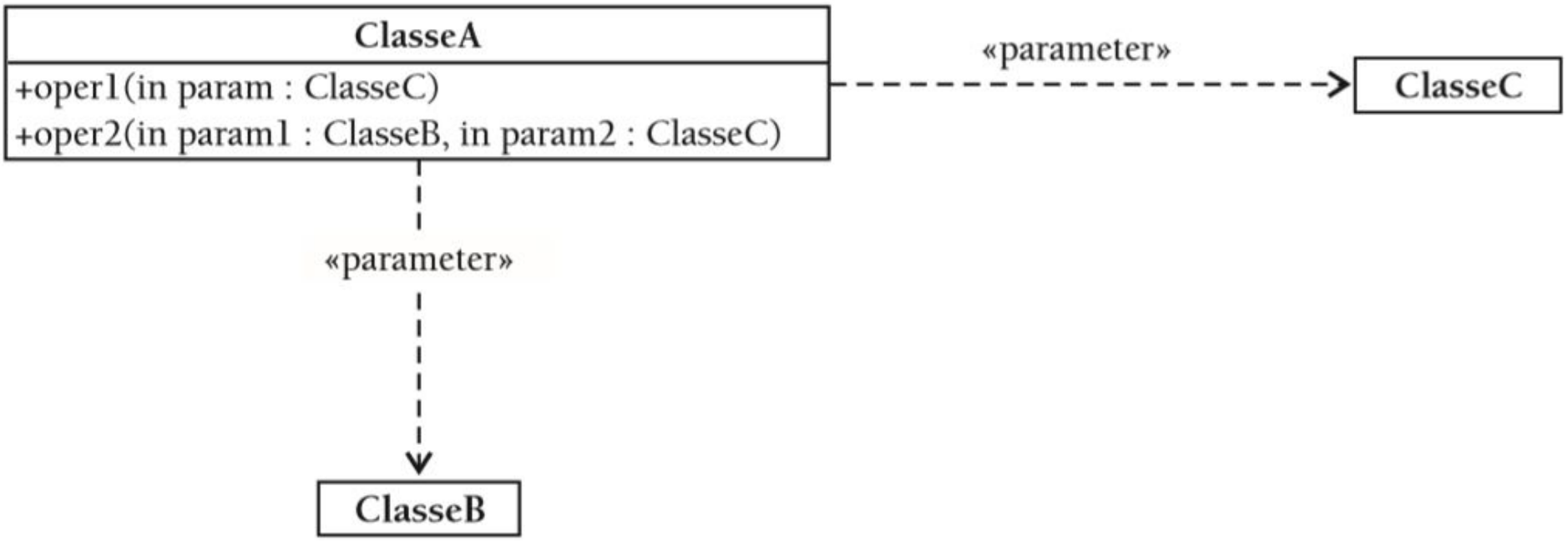
Estereótipos de Dependência

- As dependências podem ser estereotipadas para indicar o tipo de dependência existente entre as duas classes

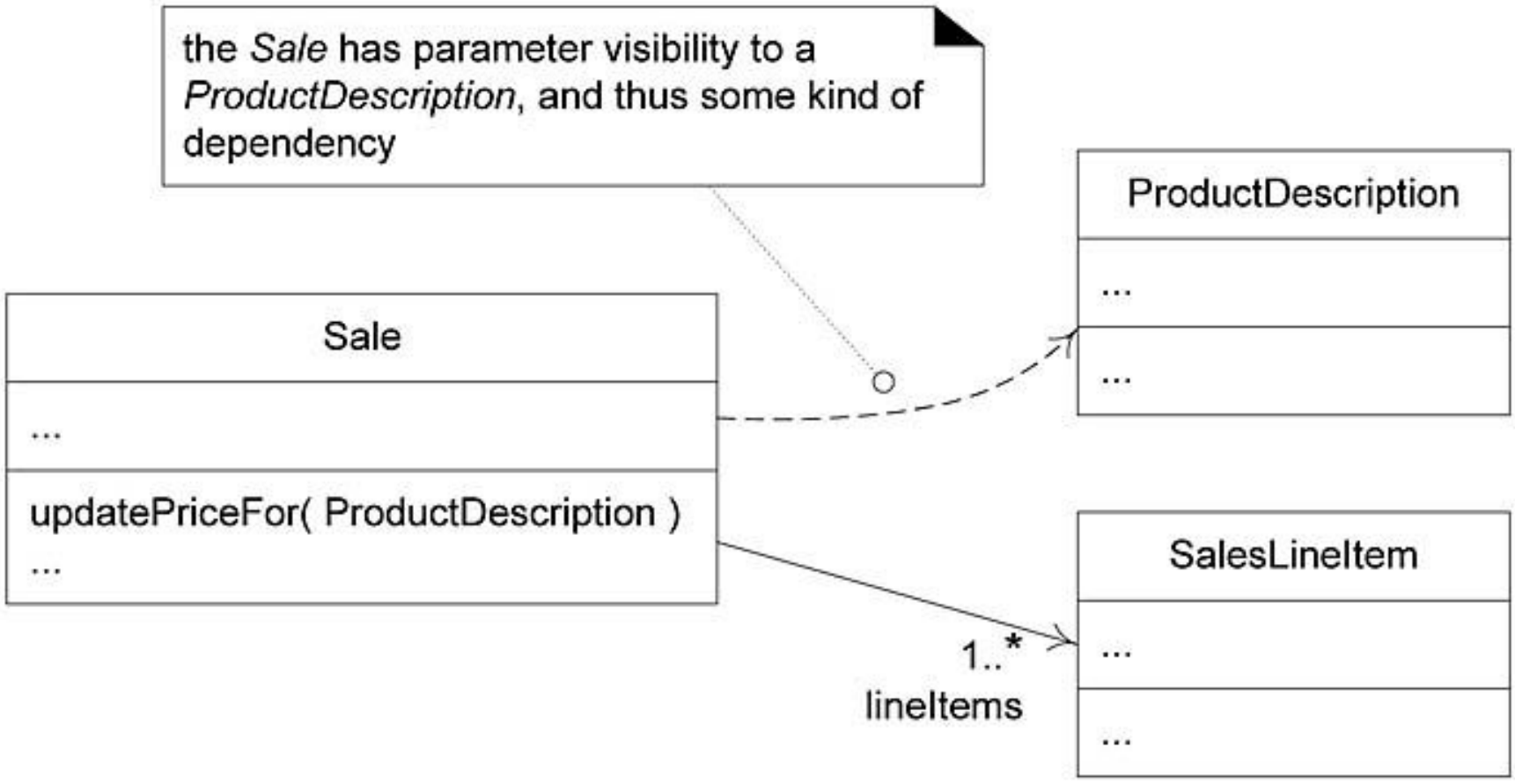
- Estereótipos pré-definidos na UML

Estereótipo	Tipo de dependência
<<global>>	Por variável global
<<local>>	Por variável local
<<parameter>>	Por parâmetro

Exemplo de Dependência (Fragmento)



Exemplo de Dependência (Fragmento)



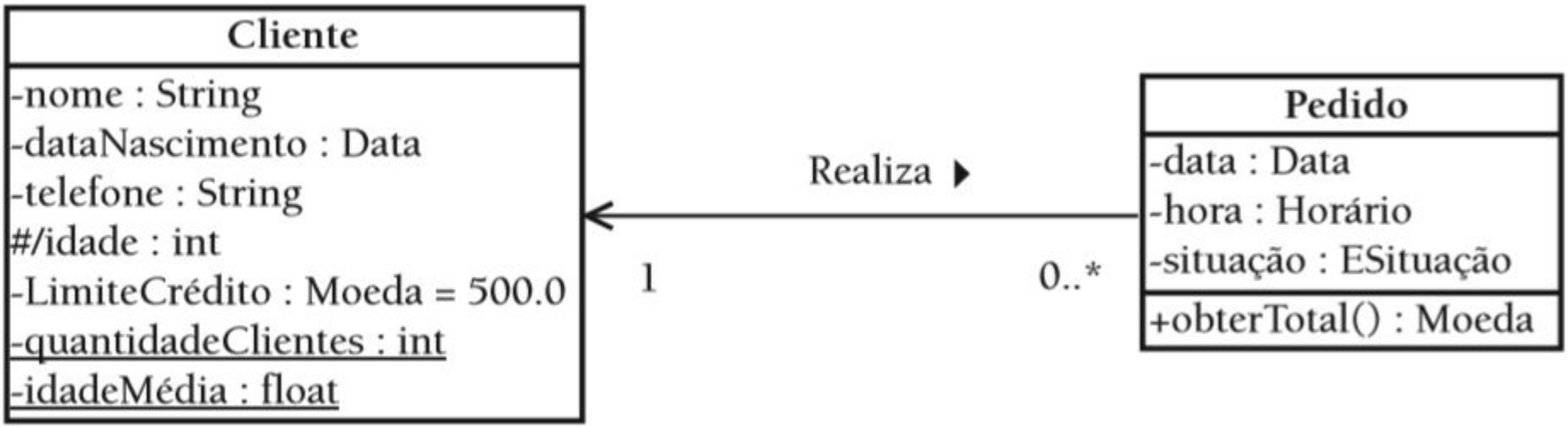
De Associações para Dependências

- Associações entre classes de domínio
  - Geralmente são dependências estruturais no projeto
- Associações entre controladores e classes de entidade
  - São bastante suscetíveis a serem transformadas em dependências não estruturais no projeto
- Associações entre classes de fronteira e controladores
  - Considerar o padrão de interação de cada classe de fronteira
  - Se é um ator humano: dependência estrutural com o controlador do caso de uso
  - Se é um ator hardware ou software: haverá interfaces para essas classes, situação em que é mais apropriado utilizar dependências não estruturais

Navegabilidade de associações

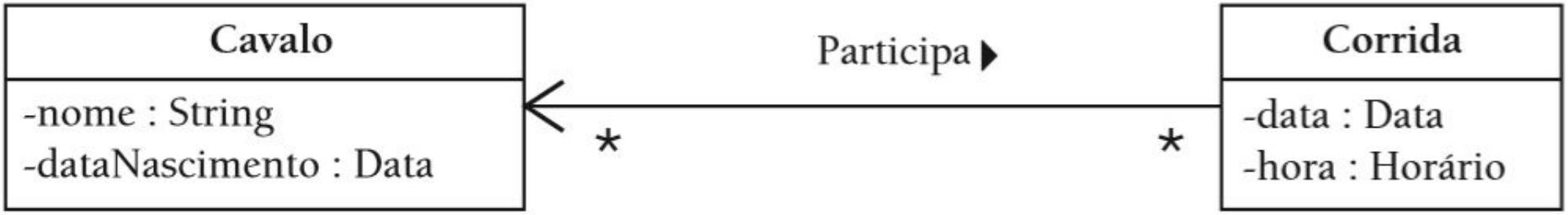
- Associação bidirecional
  - Considere as classes *C1* e *C2*
  - Cada objeto de *C1* conhece todos os objetos de *C2* aos quais está associado
  - Cada objeto de *C2* conhece todos os objetos de *C1* aos quais está associada
  - Só usar associações bidirecionais quando necessário, pois é mais difícil implementar e manter
- Associação unidirecional
  - Graficamente, há uma direção à associação, isto é: seta contínua com direção
  - A classe para a qual o sentido aponta é aquela cujos objetos não possuem visibilidade dos objetos da outra classe

Exemplo



Cada objeto Pedido conhece o seu objeto correspondente na classe cliente.

Bidirecional como Unidirecional



- Bidirecional implementada como unidirecional, sendo necessárias checagens em corridas
  - Só faz sentido se a quantidade de corridas for pequena

Um para Um

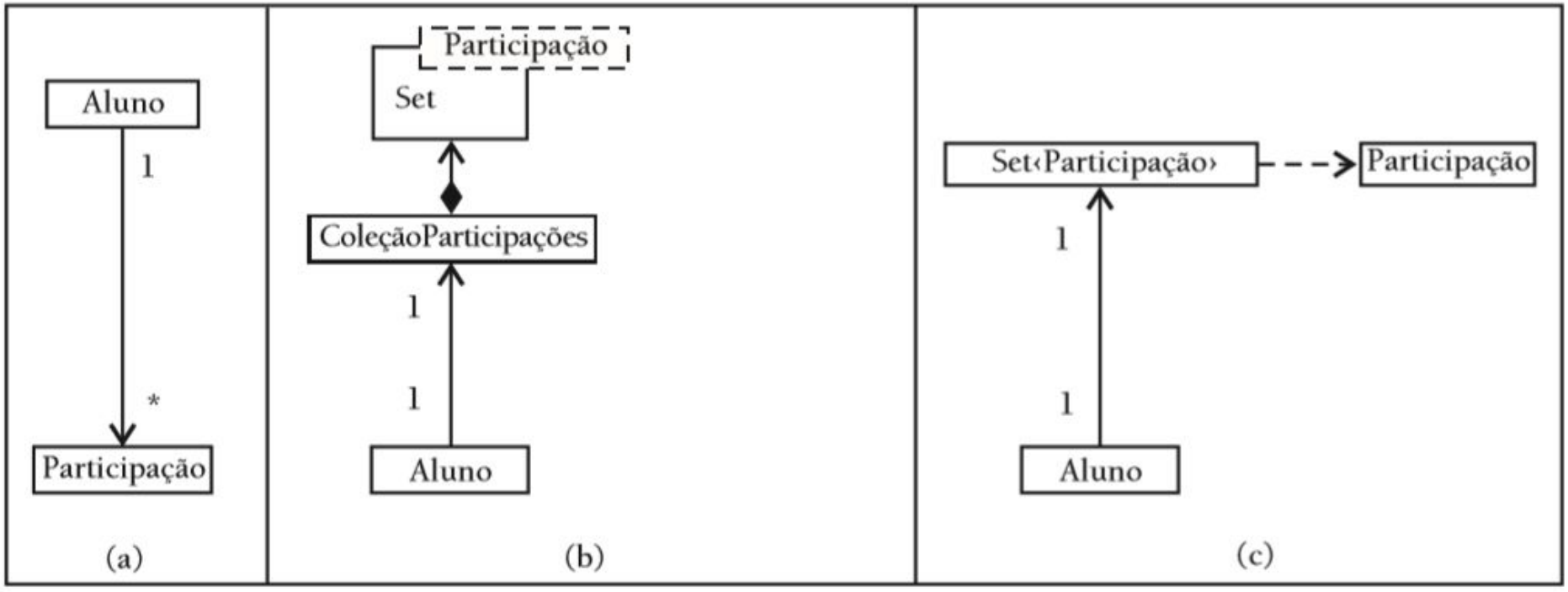
- Associações de conectividade um para um
  - Associação um para um entre *Ca* e *Cb*
  - Navegabilidade unidirecional de *Ca* para *Cb*
    - Atributo de *Cb* em *Ca*
  - Navegabilidade bidirecional entre *Ca* e *Cb*
    - Atributo de *Cb* em *Ca*
    - Atributo de *Ca* em *Cb*

Classe Parametrizada

- Associação de conectividade um para muitos ou muitos para muitos
  - Se baseia em classes que representam coleções de elementos
  - classe parametrizada, que permite representar coleções

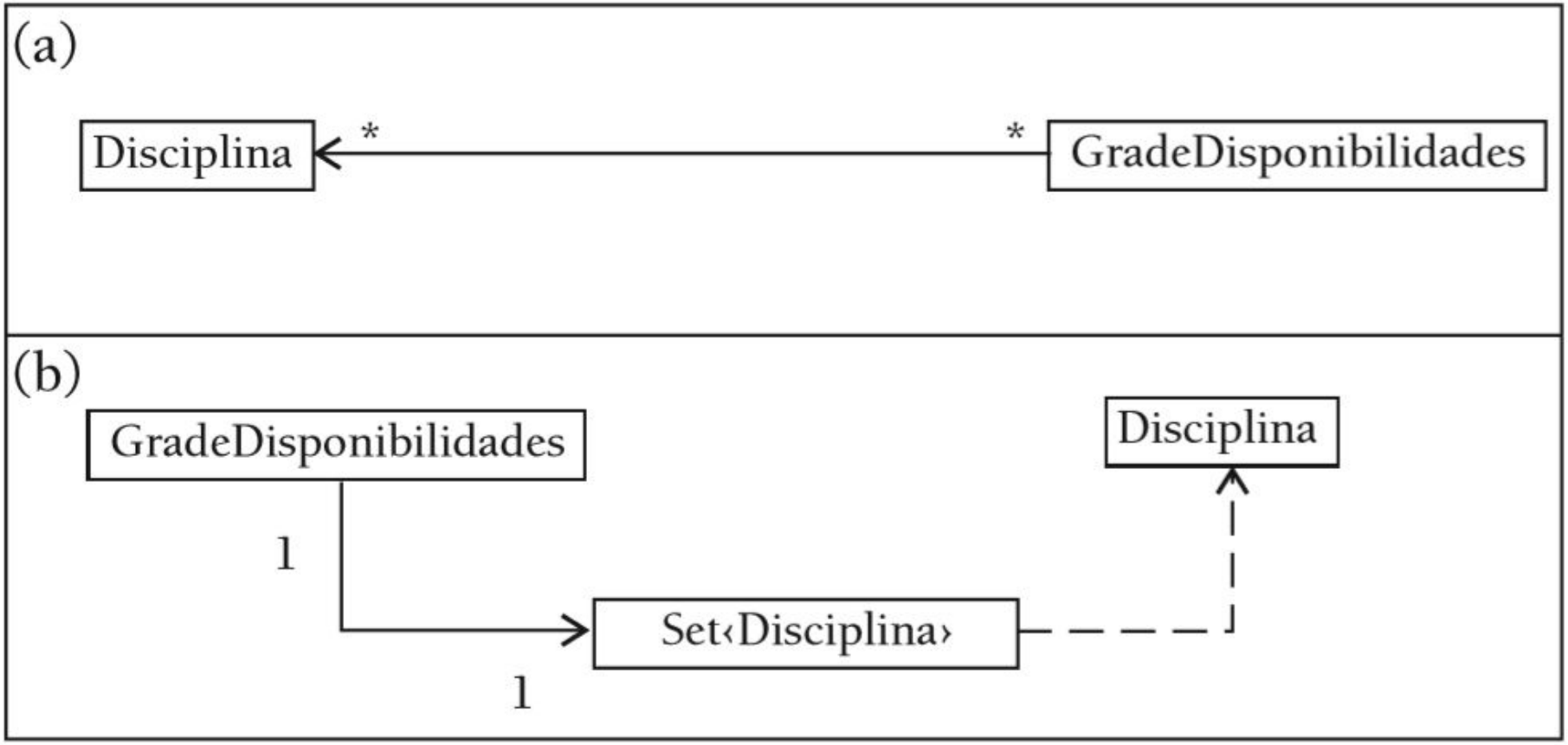
Notações possíveis na UML para uma classe parametrizada.

Um para Muitos



Formas alternativas para representar uma associação um para muitos.

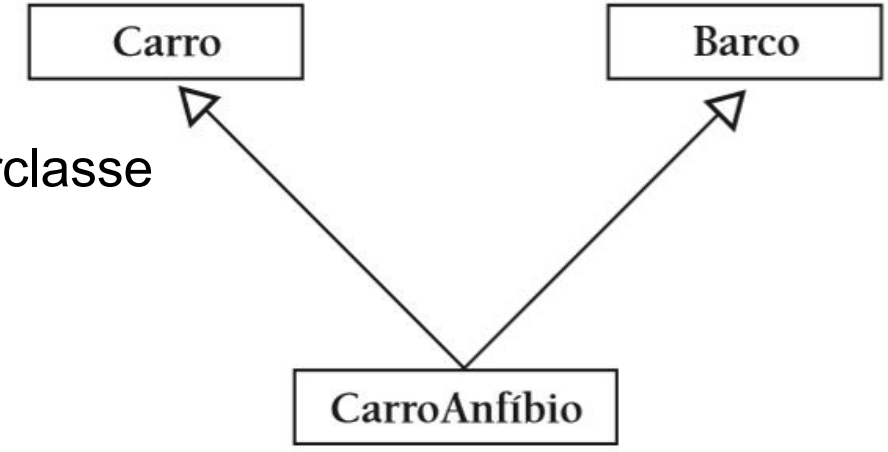
Muitos para Muitos



Associação muitos para muitos entre Disciplina e GradeDisponbilidades.

Representação de Herança em Projeto

- Herança simples
  - Uma classe tem apenas uma superclasse
- Herança múltipla
  - Uma classe tem mais de uma superclasse
- Deve-se evitar o uso de herança múltipla
  - Difícil de entender
  - Muitas linguagens não dão todo suporte à implementação (Java, C#, Smalltalk)
  - Interface é uma alternativa mais usada



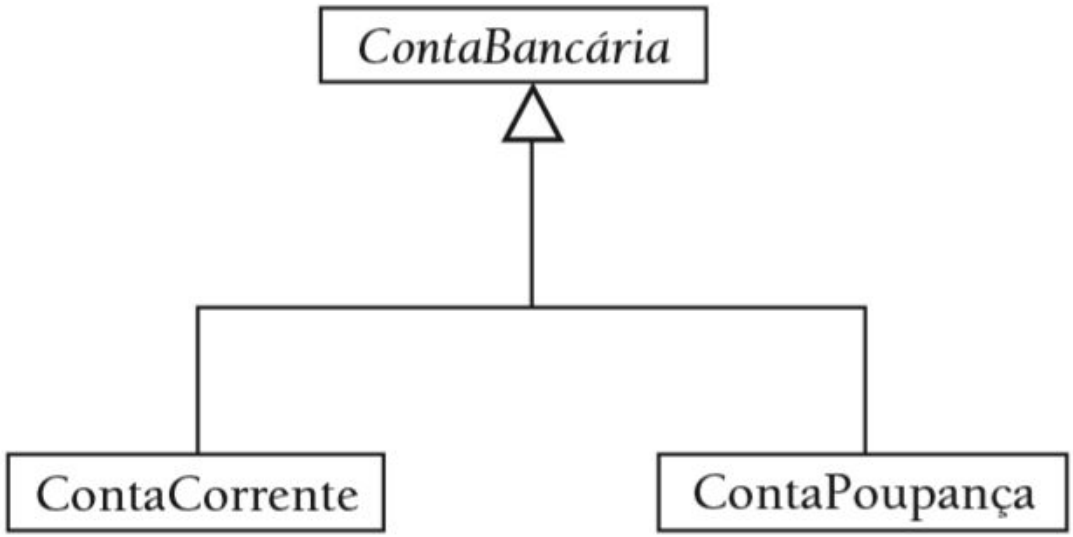
Exemplo de herança múltipla.

Herança de Implementação e de Interface

- Serviço
  - Especificação (o que o serviço faz) e método (modo como realizar o serviço)
  - Uma especificação pode corresponder a diversos métodos
- Herança de Implementação
  - Subclasse herda as especificações definidas na interface de um ancestral e se compromete a implementar essa interface
- Herança de Interface
  - Envolve os conceitos de classe abstrata e interface

Classe Abstrata

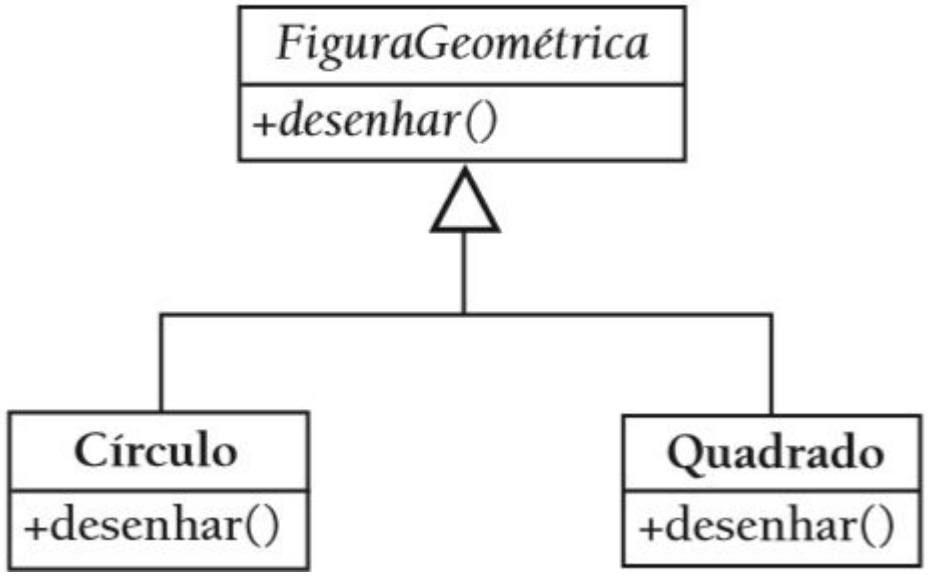
- Classes abstratas não possuem instâncias
  - São usadas apenas em hierarquia



O nome da classe abstrata fica em itálico ou a etiqueta <<*abstract*>>

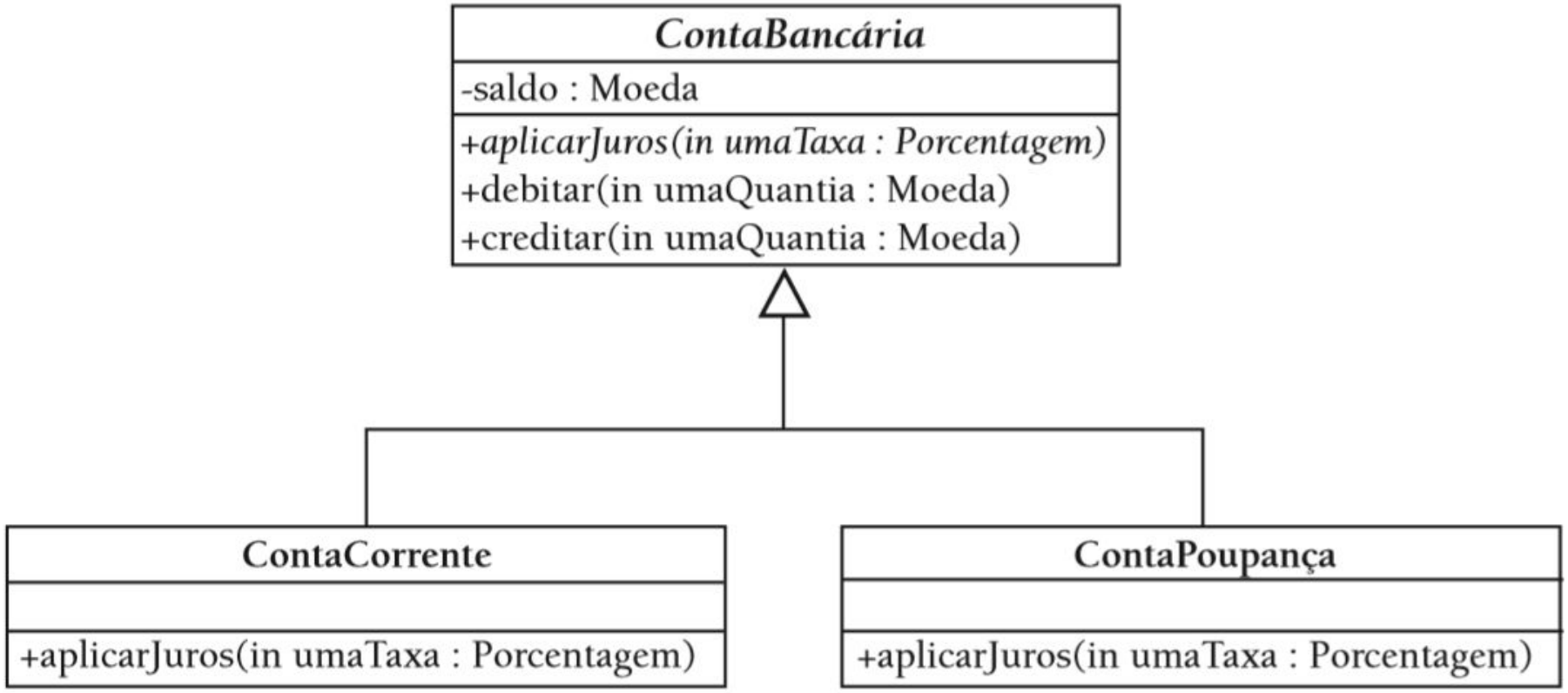
Operações abstratas

- Classes abstratas possuem ao menos uma operação abstrata
  - Se uma classe herda uma operação abstrata de uma classe abstrata e não implementa a operação, então ela se torna abstrata



O nome da operação abstrata fica em itálico

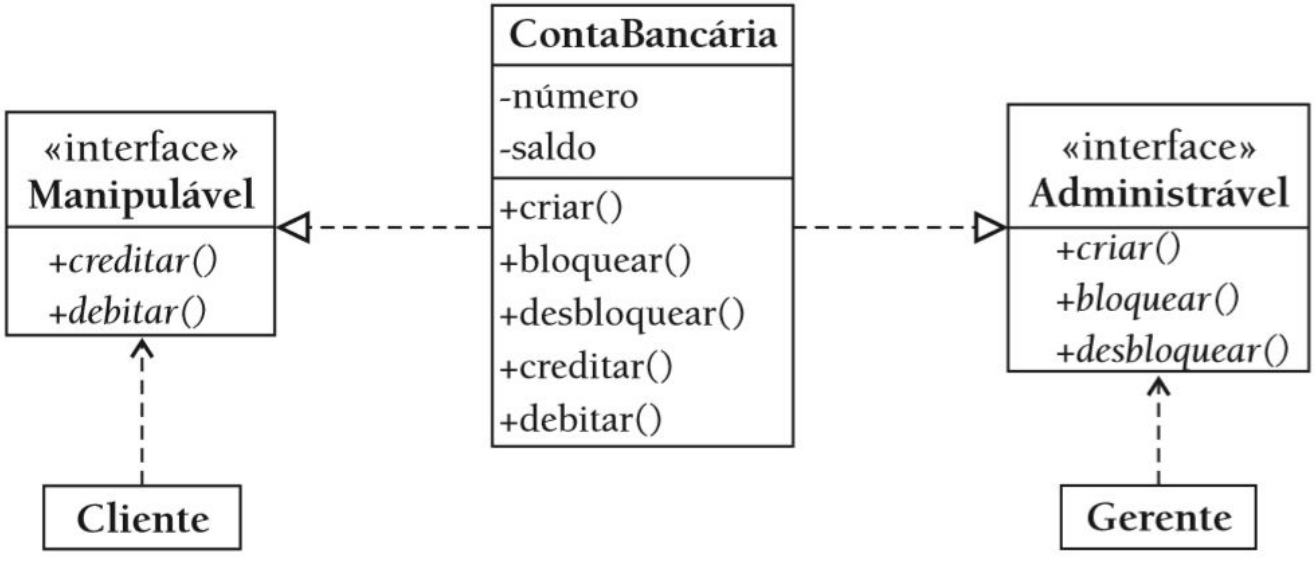
Operações Polimórficas



Interfaces

- Podemos definir os seguintes objetivos do conceito de interface:
  - Capturar semelhanças entre classes não relacionadas sem forçar relacionamentos entre elas
  - Declarar operações que uma ou mais classes devem implementar
  - Revelar as operações de um objeto, sem revelar a sua classe
  - Facilitar o desacoplamento entre elementos de um sistema

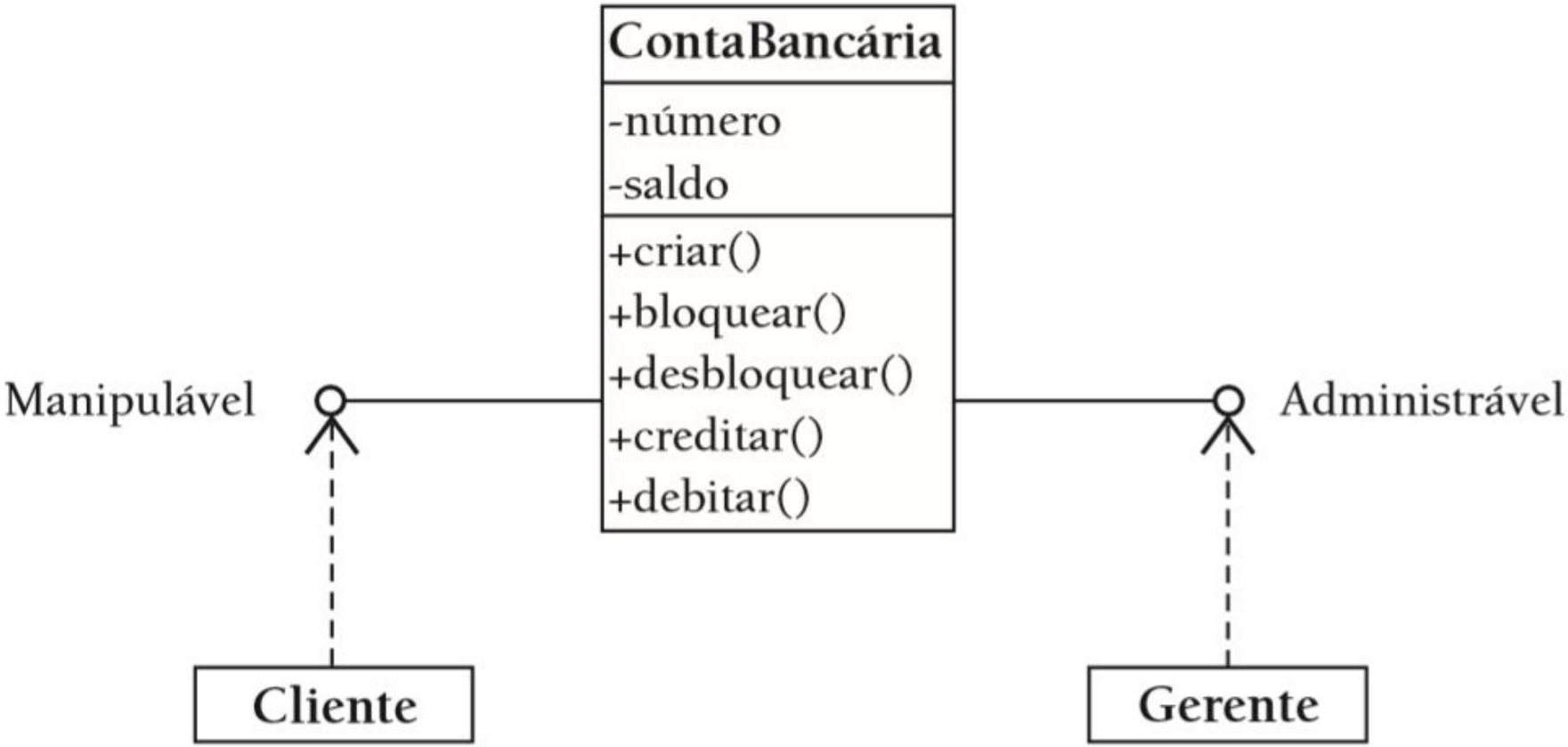
Exemplo de Interface



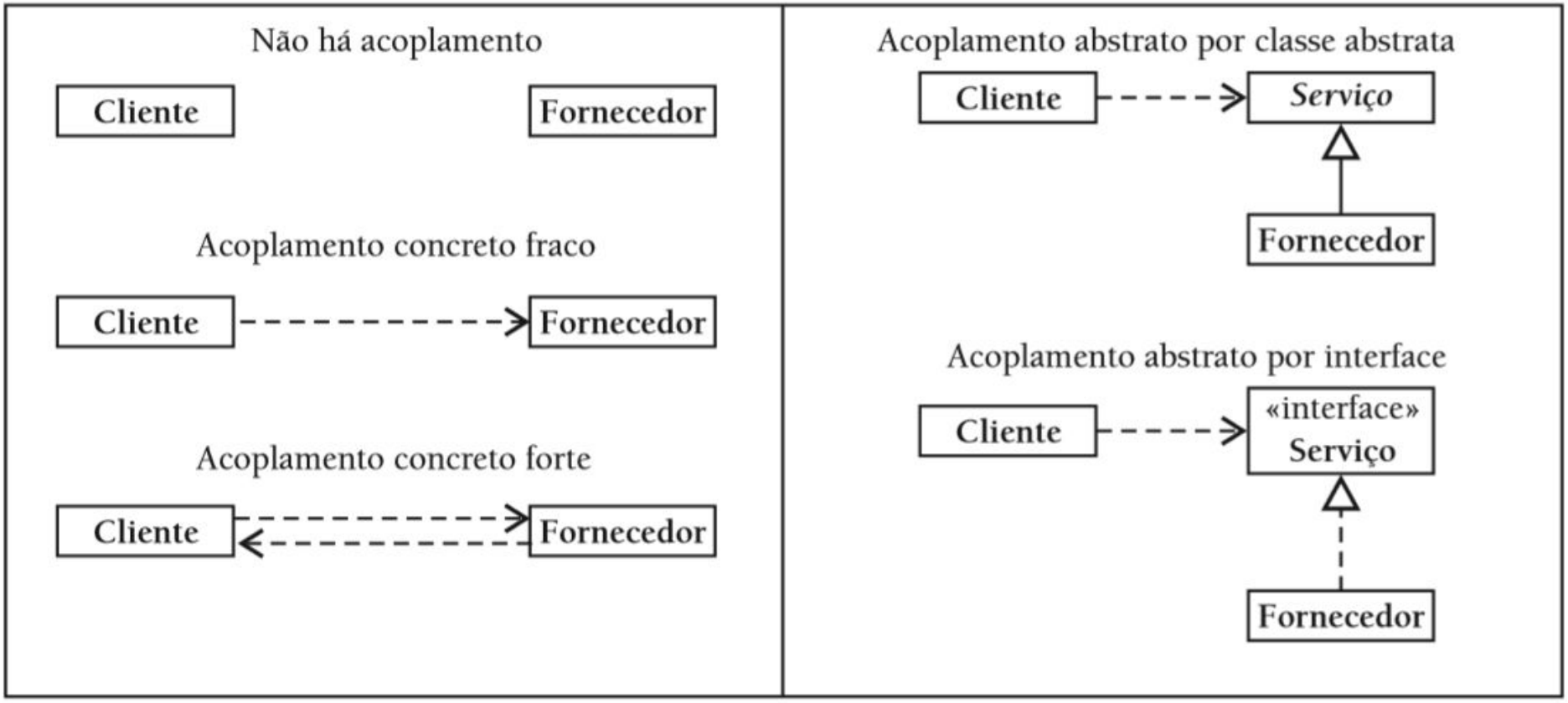
- O símbolo da seta fechada com linha pontilhada indica implementação, os métodos são implementados em *ContaBancária*

- Para as classes Cliente e Gerente, que utilizam as interfaces que *ContaBancária* realiza, tudo se passa como se estivessem utilizando a própria classe
  - Porém, essas classes somente visualizam as operações que fazem sentido para elas

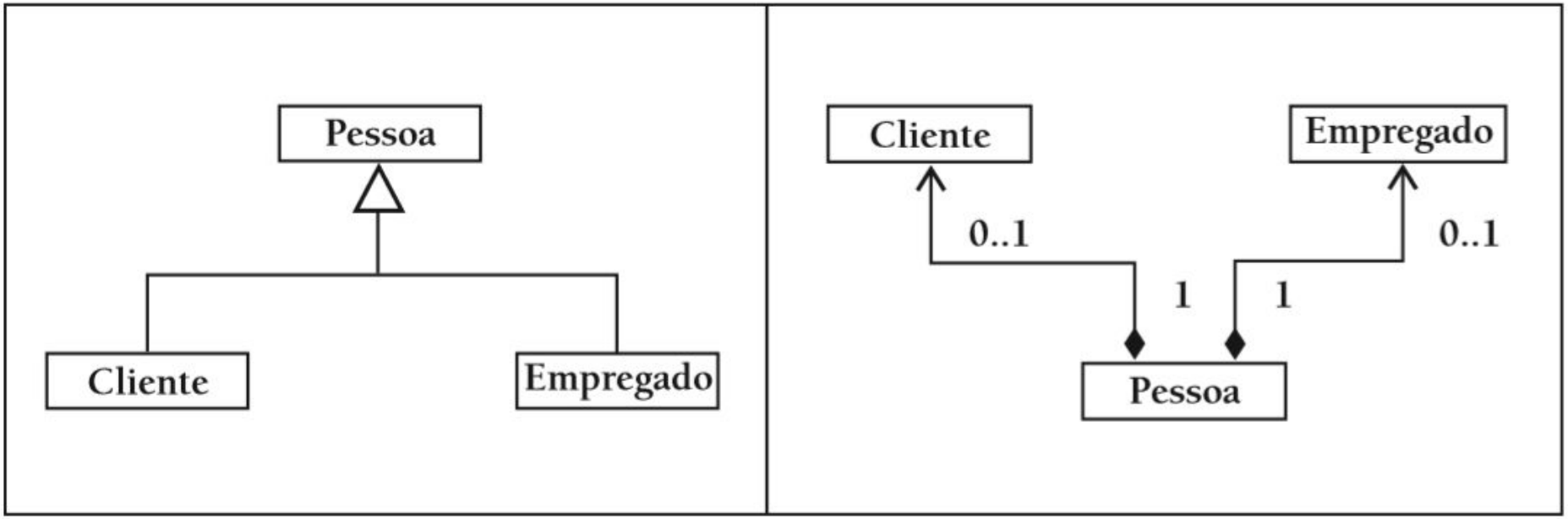
Exemplo em Formato Simplificado



Acoplamentos Concreto e Abstrato



Classificação Dinâmica



Referências

BEZERRA, E.; Princípio de Análise e Projeto de Sistemas com UML, Rio de Janeiro, Elsevier, 2007. **(Capítulo 8)**

LARMAN, C.; Utilizando UML e Padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo, Porto Alegre, Bookman, 2007. **(Capítulo 16)**

GUEDES, Gilleanes T. A. UML 2: uma abordagem prática. 2. ed. São Paulo: Novatec, c2011. 484 p. ISBN 9788575222812 **(Capítulo 4)**

Projeto de Software

Prof. Dr. Lesandro Ponciano

https://orcid.org/0000-0002-5724-0094