Fatec
Guaratinguetá

Tecnologia em Análise e Desenvolvimento de Sistemas

Engenharia de Software II

UML – DIAGRAMA DE CLASSES

Prof. Claudemir Santos Pinto
claudemir.santos2@fatec.sp.gov.br

UML – Unified Modeling Language

- **Diagrama de Classes:**

É usado para descrever a estrutura estática de classes no sistema, permitindo definir os **atributos**, **métodos** (operações) e **relacionamento** entre as classes.

Apresenta uma visão estática da organização das classes, definindo sua estrutura lógica.

É um dos diagramas mais populares e serve como base para outros diagramas.

Basicamente, descreve o que deve estar presente no sistema modelado

UML – Unified Modeling Language

- **Diagrama de Classes:**

  Serve para modelar o vocabulário de um sistema, é construído e refinado ao longo das várias fases do desenvolvimento do software (**análise, projeto e implementação**)

  Também serve para:

  - Especificar colaborações
  - Especificar esquemas lógicos de bases de dados
  - Especificar visões (estrutura de dados de formulários, relatórios, etc.)

UML – Unified Modeling Language

Classes, Atributos e Métodos

- Uma classe é uma representação de um item do mundo real, físico ou abstrato, na forma de um tipo de dados personalizado.
- As classes possuem estruturas internas chamadas de **atributos** e **métodos**.
- **Atributos** são usados para armazenar os dados dos objetos de uma classe.
- **Métodos** são operações ou funções que a instância de uma classe pode executar.
- Uma instância de uma classe é chamada **Objeto.**

Classe, atributos e métodos - exemplo

Classe: Pessoa

Atributos: Nome, Altura, Idade, Peso

Métodos: Andar, Comer, Falar, Estudar, Dormir, Trabalhar

Objeto da Classe (instância):

Atributos:

- Nome: João
- Altura: 1,80
- Idade: 30
- Peso: 80

(é um exemplar específico da classe)

* Os métodos estarão disponíveis, ainda que não sejam usados por todos os objetos

Representação de uma classe

Representamos uma classe usando um retângulo dividido em três compartimentos:

Nome: inclui o nome e estereótipo da classe

Atributos: lista de atributos da classe no formato:
nome: **tipo** ou **nome**:**tipo**=**valor** (quando houver um valor padrão)

Métodos: lista de operações da classe no formato:
método(parâmetro):**tipo_retorno** (void se não retornar nada)

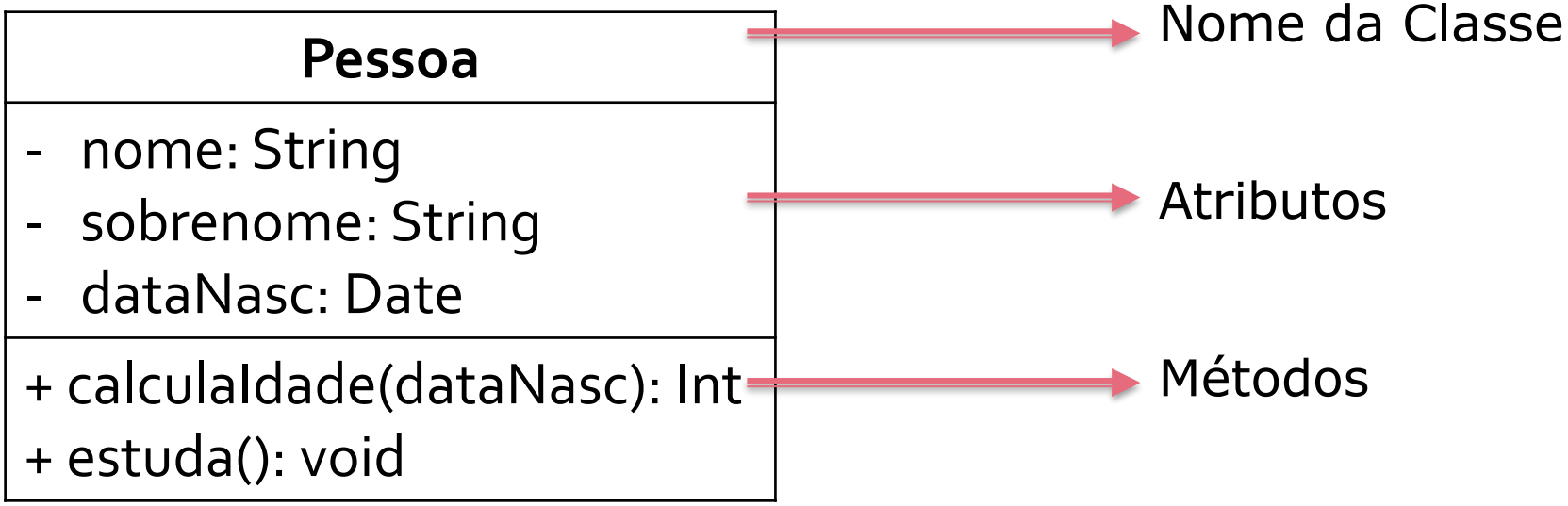
Visibilidade dos atributos/métodos

Representamos a visibilidade dos atributos e operações usando os modificadores de acesso a seguir:

+ Público (visível em qualquer classe de qualquer pacote)
# Protegido (visível para classes e subclasses)
- Privado (visível somente para a classe)
~ Pacote (visível para classes do mesmo pacote)

Representação de uma classe

Exemplo: representando uma classe pessoa, que contém os atributos nome, sobrenome e dataNasc, além do método calculaIdade:



get: pegar valor do atributo. Ex: getNome(): string
set: alterar valor do atributo. Ex: setNome(valor:string): void

Relacionamento entre Classes

Um relacionamento é uma conexão entre as classes.
Existem vários tipos de relacionamento possíveis:

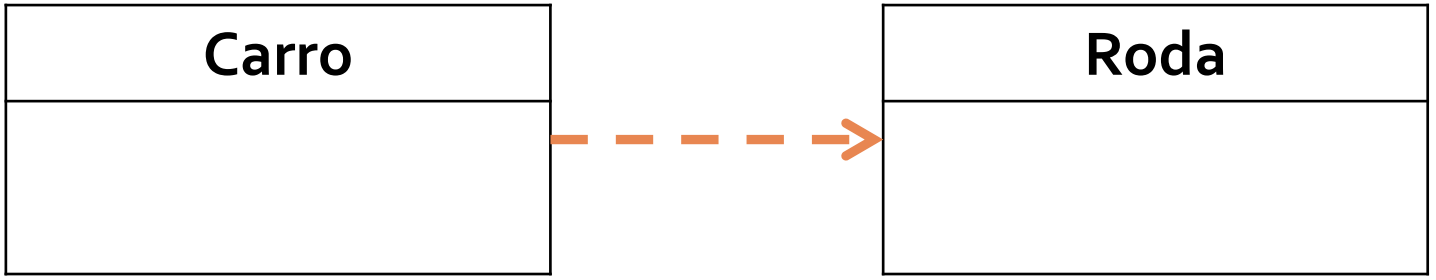
- Dependência
- Agregação
- Associação
- Composição
- Generalização/Especialização

Cada um desses relacionamentos possui uma representação gráfica específica

Relacionamento de Dependência

Dependência fraca, usualmente transiente, que ilustra que uma classe usa informações e serviços de outra classe em algum momento, dependendo dela.

Do tipo "Classe A *depende* de Classe B":



Multiplicidade

É usada para determinar o número mínimo e máximo de objetos envolvidos na associação de cada lado, e também pode especificar o nível de dependência entre os objetos.

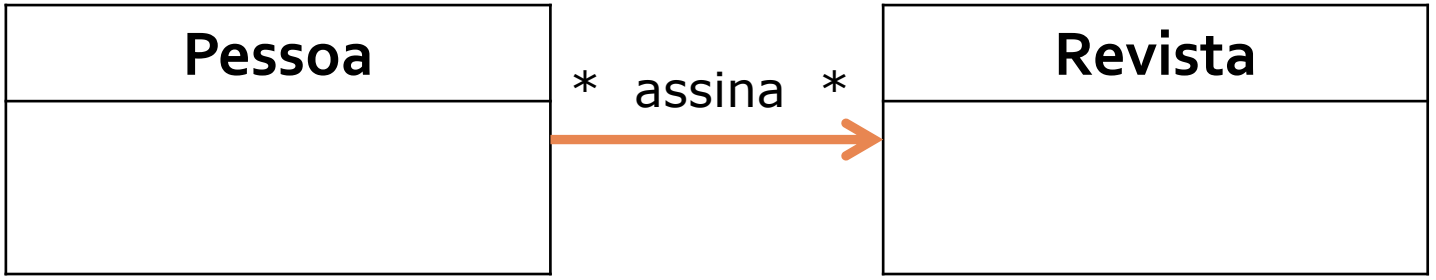
Multiplicidade	**Significado**
0..1	No mínimo zero e no máximo um. Indica não-obrigatoriedade do relacionamento.
1..1	UM e somente UM. UM objeto de uma classe se relaciona com UM objeto da outra classe (não é obrigatório exibir essa multiplicidade)
0..*	Mínimo zero e no máximo muitos
1..*	Mínimo 1 e no máximo muitos
*	Muitos
2..7	Mínimo 2 e no máximo 7.

Relacionamento de Associação

Mais forte que a Dependência, indica que a classe mantém uma referência a outra classe ao longo do tempo. As associações podem conectar mais de duas classes.

Do tipo "Classe A *tem uma* Classe B"

A seta representa a Navegabilidade, que identifica o sentido que a informação é transmitida entre os objetos das classes tradicionais. A Navegabilidade não é obrigatória nesse diagrama.



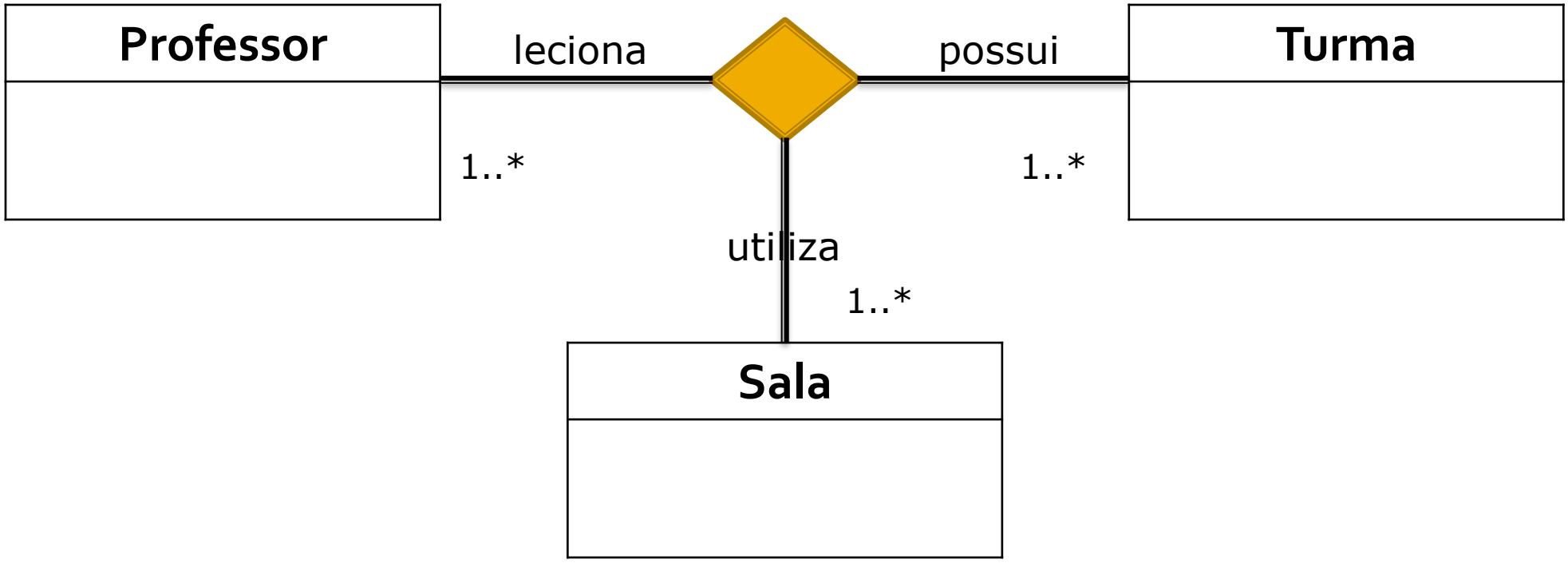
Pessoa realiza algum método na classe revista, mas não vice-versa.

Quando a associação possibilita navegação nos dois sentidos, não é necessário por as setas, basta uma linha simples.

A associação pode ter um nome e possui multiplicidade

Associação ternária

Conecta objetos de três classes. Um losango indica um ponto de convergência (conexão) das classes envolvidas.

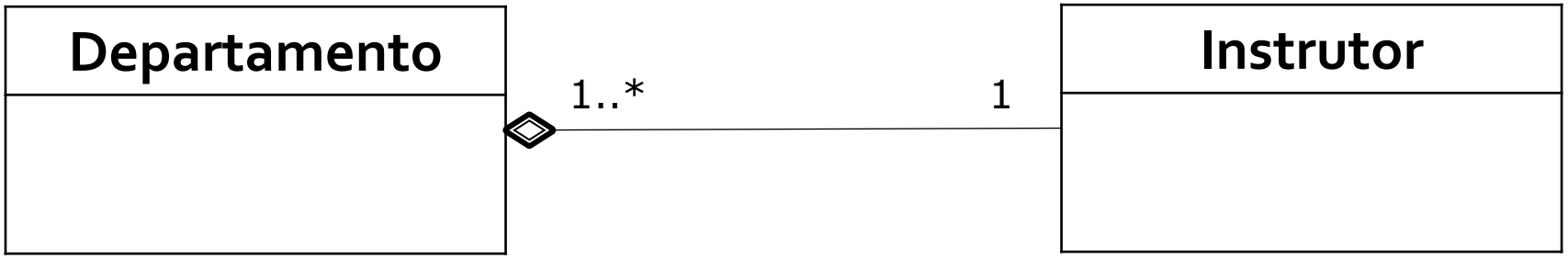


Na prática podemos ter associações **N-árias,** conectando quaisquer números de objetos de classes

Relacionamento de Agregação

Relacionamento mais específico que uma associação, indica que uma classe é um contêiner ou uma coleção de outras classes. As classes contidas não dependem do contêiner, assim, quando o contêiner é destruído, as classes continuam existindo.

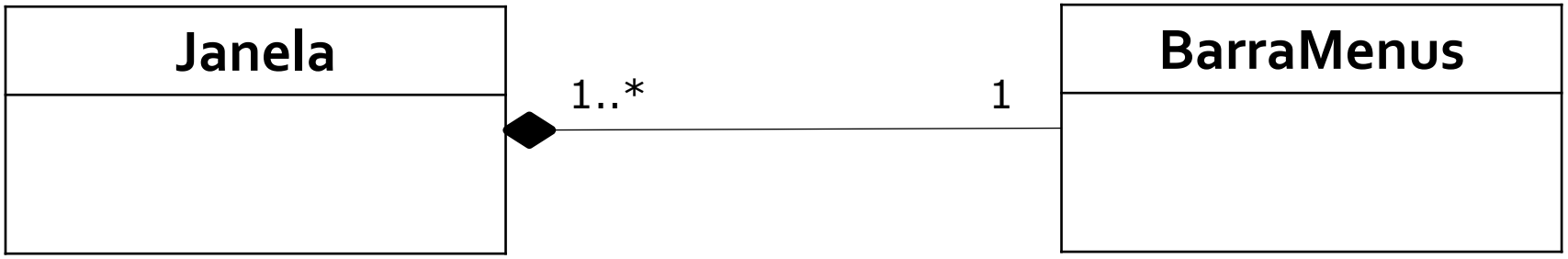
Do tipo "Classe A *possui* Classe B"



Relacionamento de Composição

Variação mais específico da agregação, este relacionamento indica uma dependência do ciclo de vida forte entre as classes, de modo que quando um contêiner é destruído, seu conteúdo também o é.

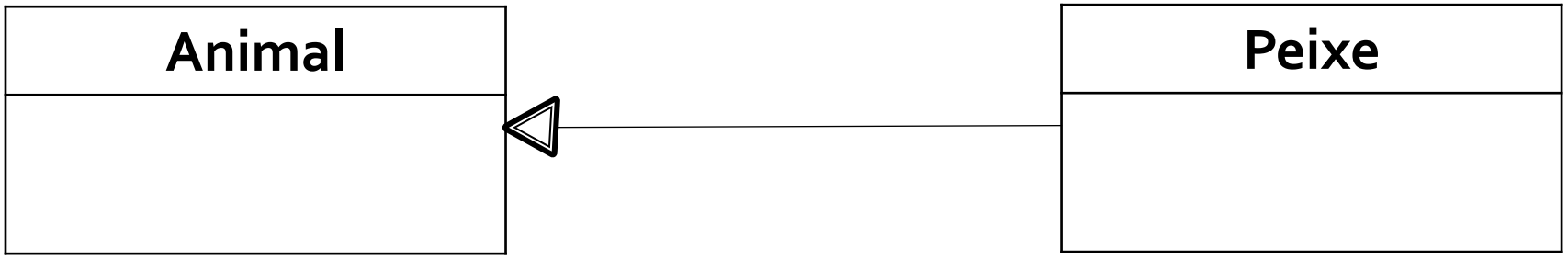
Do tipo "Classe B *é parte da* Classe A"



Relacionamento de Generalização / Especialização

Relacionamento entre itens gerais (super-classes / classes-mãe) e tipos mais específicos desses itens (sub-classes / classes-filhas). Representa a herança entre as classes.

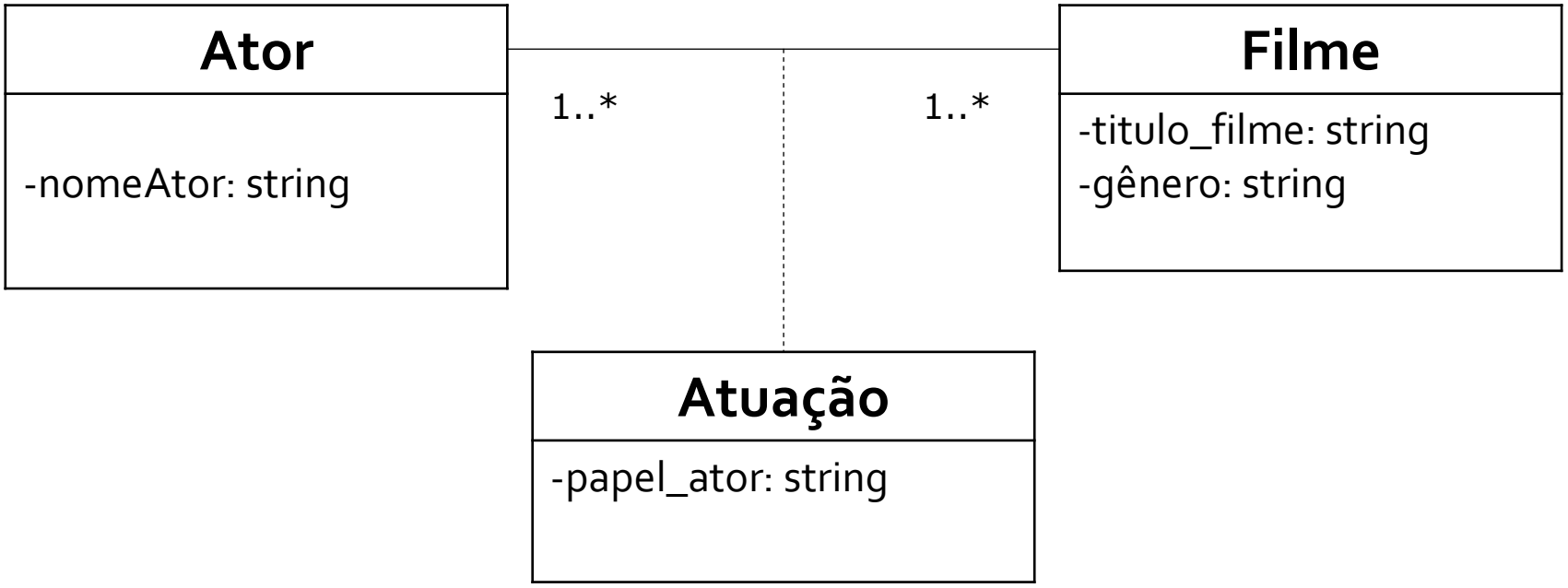
Do tipo "Classe B *é um tipo de* Classe A"



A classe filha herda propriedades da classe mãe, principalmente, atributos e operações, e pode possuir seus próprios

Classe associativa

São produzidas quando ocorrem associações com multiplicidade muitos (*) em todas as extremidades. No geral , existem atributos da associação que não podem ser armazenadas em nenhumas das classes envolvidas



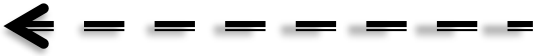
Resumo da notação

Associação 

Agregação 

Composição 

Herança 

Dependência 

Boas práticas

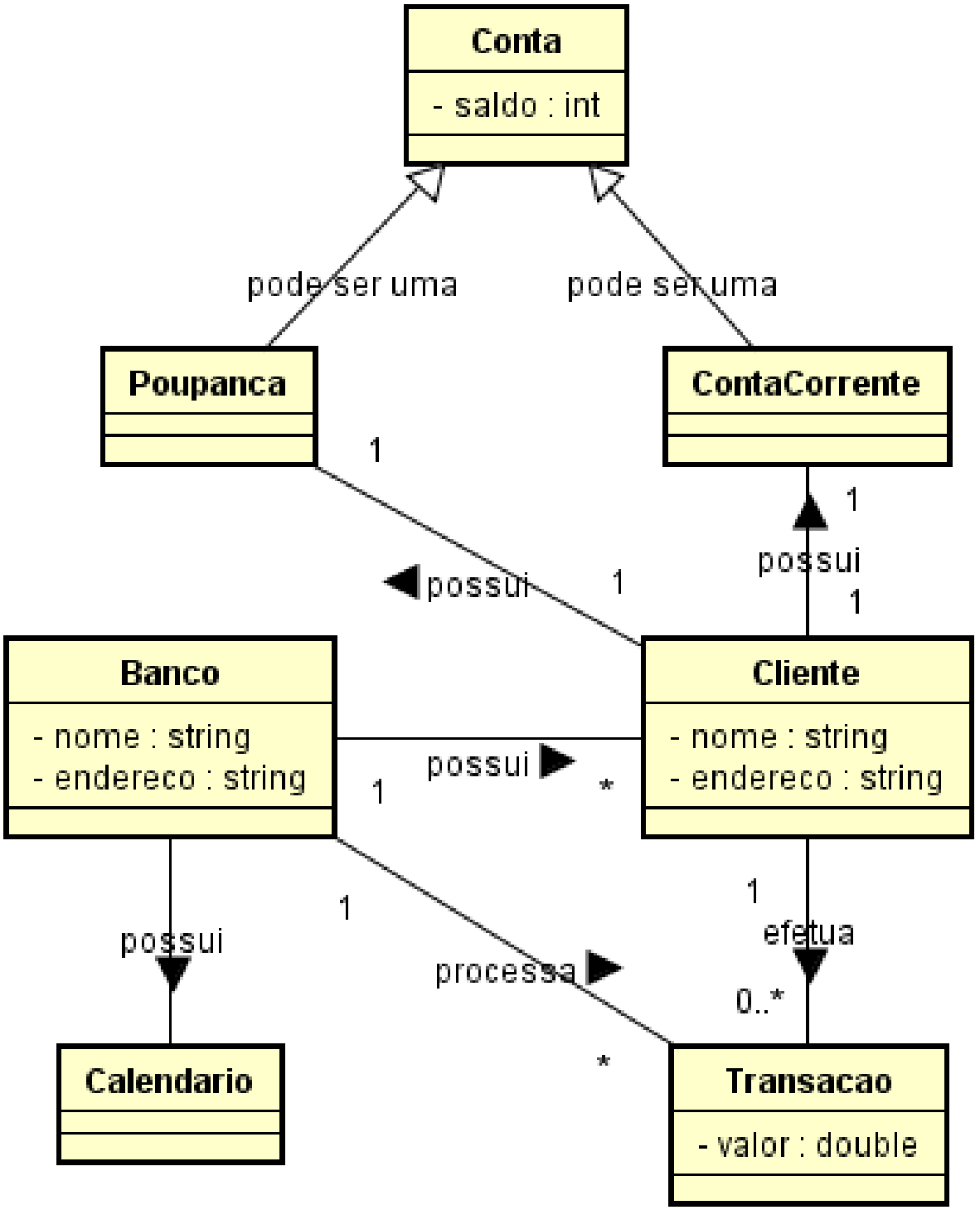
- O nome da classe deve ser significativo, descrevendo um aspecto real do sistema
- Os relacionamentos entre os elementos devem ser identificados antes de criar o diagrama
- Devem ser especificados os atributos e operações de cada classe
- Sempre que necessário acrescente anotações para ajudar a definir aspectos das classes ou seus relacionamentos

Perspectivas

Um diagrama de classes pode oferecer três perspectivas, cada uma para um tipo de observador diferente. São elas:

- ✓ Conceitual
  - Representa os conceitos do domínio em estudo.
  - Perspectiva destinada ao cliente.

Exemplo
Perspectiva
Conceitual

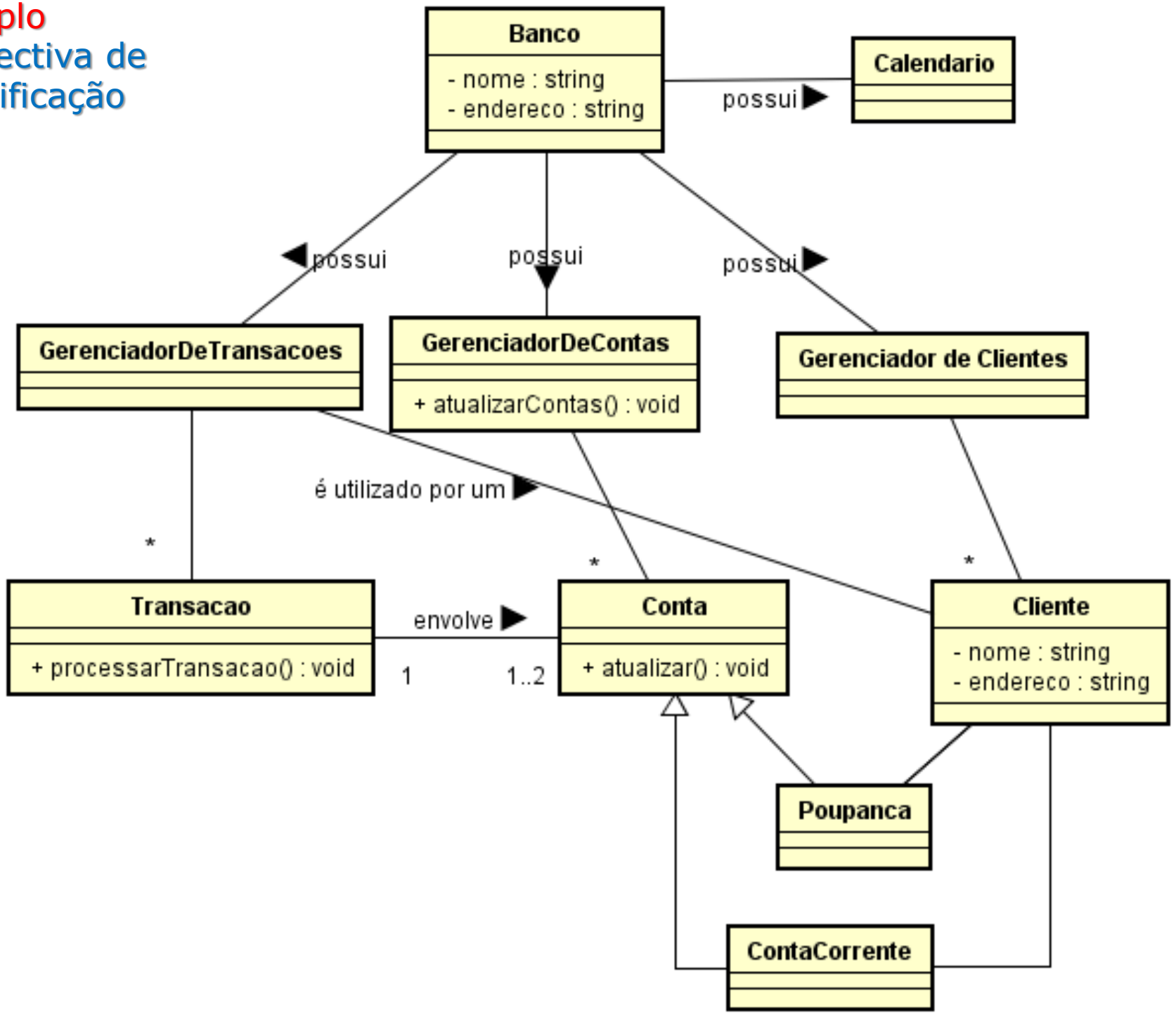


Perspectivas

Um diagrama de classes pode oferecer três perspectivas, cada uma para um tipo de observador diferente. São elas:

- ✓ Especificação
  - Tem foco nas principais interfaces da arquitetura, nos principais métodos, e não como eles irão ser implementados.
  - Perspectiva destinada as pessoas que não precisam saber detalhes de desenvolvimento, tais como gerentes de projeto.

Exemplo
Perspectiva de Especificação



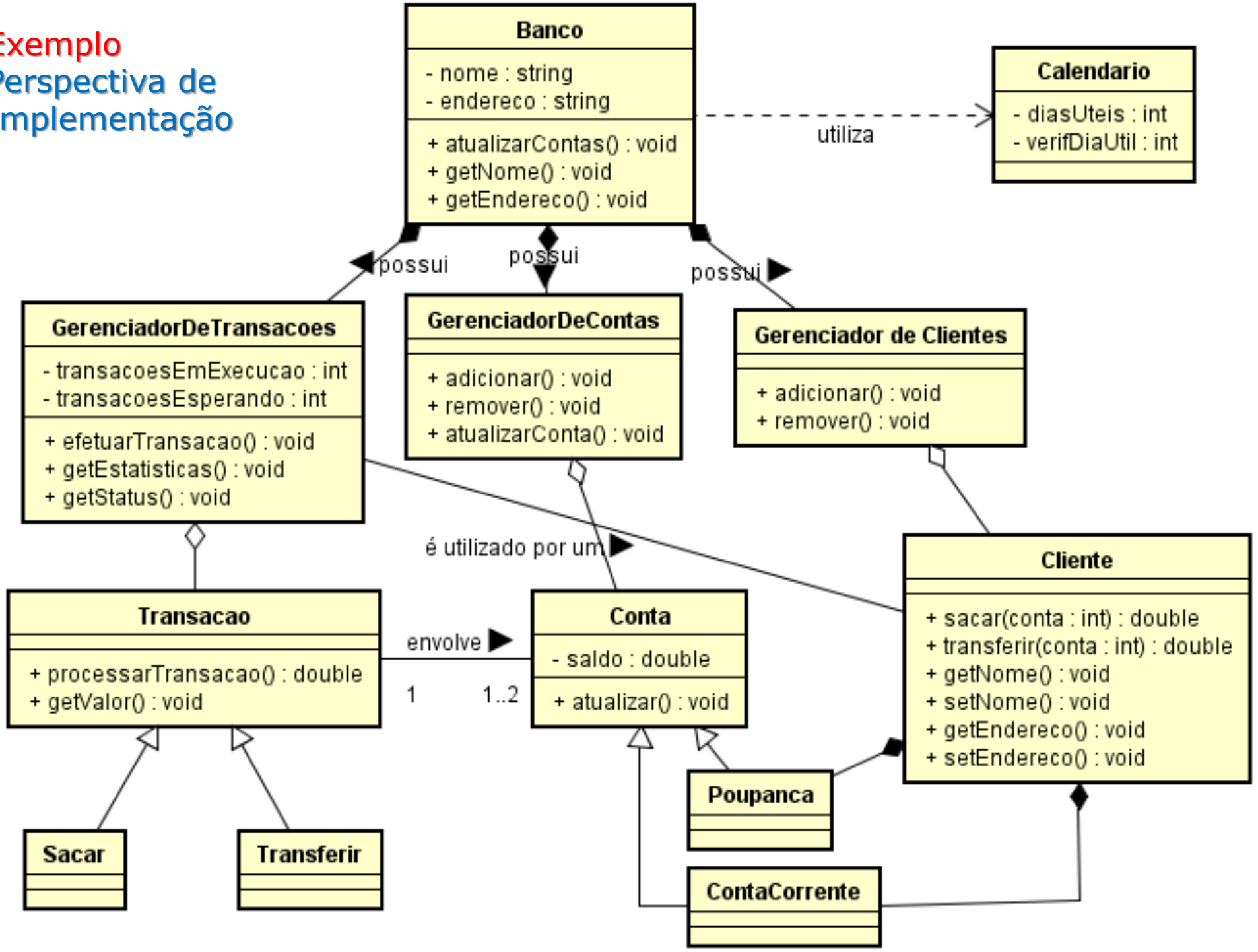
Perspectivas

Um diagrama de classes pode oferecer três perspectivas, cada uma para um tipo de observador diferente. São elas:

✓ Implementação

- Aborda vários detalhes de implementação, tais como navegabilidade, *tipo* dos atributos, etc.
- Perspectiva destinada ao time de desenvolvimento.

Exemplo
Perspectiva de Implementação



Exemplo - cenário

- Maria é uma professora e está organizando um passeio com seus alunos. Ela precisa guardar, para cada aluno, nome, turma e quem é o responsável.

- Cada responsável possui uma ficha com seu nome, telefone fixo, telefone celular e endereço.

- Todo passeio realizado pela escola possui data, horários de saída e chegada, motorista e o local a ser visitado

- É preciso saber que alunos foram em cada passeio

Exemplo: passeio escolar