Modelagem de Dados

Diagrama de Classe

Diagramas de classes, são os diagramas encontrados com maior frequência na modelagem de sistemas orientados a objetos. Diagramas de classes, mostram um conjunto de classes, interfaces, colaborações e seus relacionamentos.

Modelagem de Dados

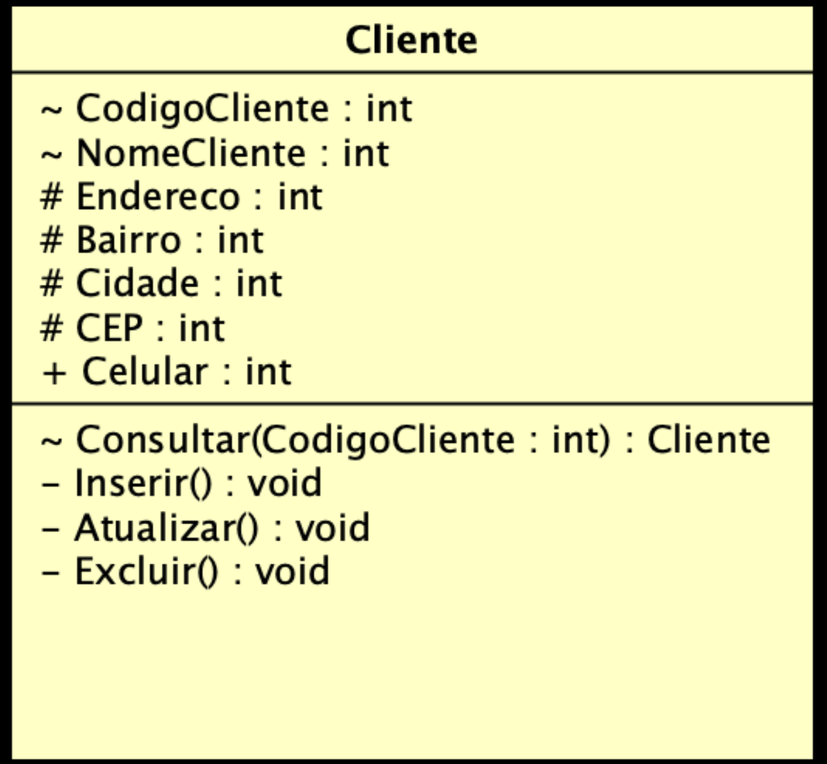
Diagrama de Classe

- Quando modelamos um sistema dentro do paradigma da orientação a objetos, este é organizado em termos de classes de objetos.
- Representa a modelagem de classes, interfaces e seus relacionamentos
- Representa sua organização em pacotes.
- Na fase de construção ou final da definição, o diagrama apresenta maior detalhamento do que o produzido na fase inicial de definicão.

Modelagem de Dados

Diagrama de Classe

		Visibilidade
public	+	fora da classe
protected	#	dentro da hierarquia da classe
private	-	dentro da classe
package	~	dentro do pacote



Modelagem de Dados

Diagrama de Classe

Relacionamentos entre pacotes

Associações de uso e acesso:

- Import (public)
  - Ex: `import java.util.ArrayList;`
- Access (private)
  - Ex: `package model;`

Modelagem de Dados

Diagrama de Classe

Relacionamentos entre classes

- Dependência
- Associação
  - Agregação
  - Composição
- Generalização

Modelagem de Dados

Diagrama de Classe | Dependência | Associação | Generalização

É o relacionamento mais simples entre classes/objetos. A dependência indica que um objeto depende da especificação de outro objeto.

Por especificação, podemos entender a interface pública do objeto (seu conjunto de métodos públicos).

Em um relacionamento de dependência, um objeto é dependente da especificação de outro objeto. Se a especificação mudar, você precisará atualizar o objeto dependente.

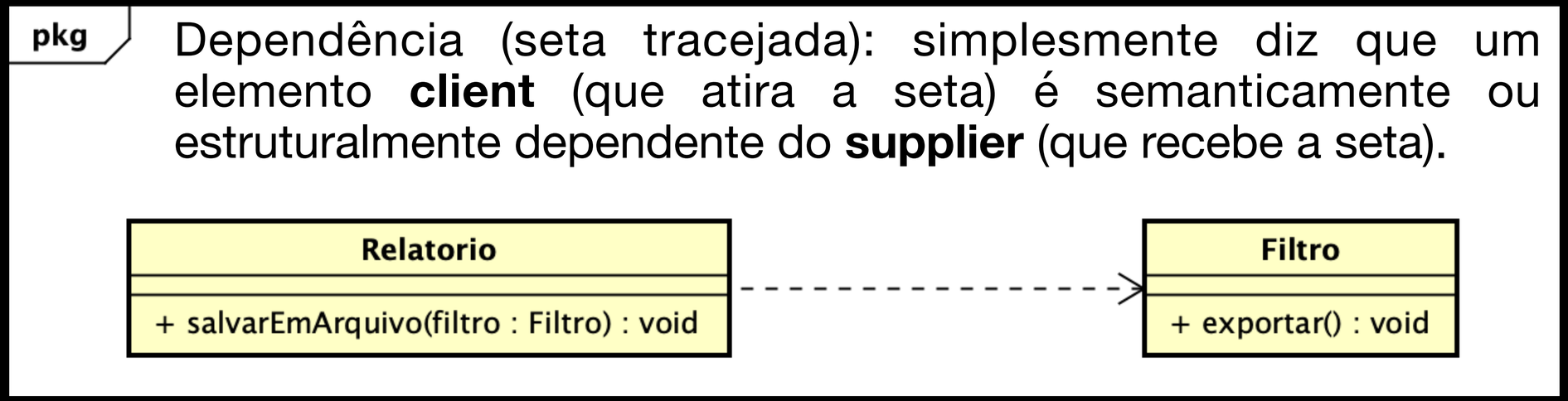
Através da POO, você sempre deve tentar minimizar o máximo possível as dependências. Entretanto, é impossível eliminar todas as dependências entre os objetos.

Modelagem de Dados

Diagrama de Classe | Dependência | Associação | Generalização

Quando modelar dependências?

Normalmente, você modela dependências quando quer mostrar que um objeto usa outro.



Um lugar comum onde um objeto usa outro é através de um argumento de método. Por exemplo, o método `salvarEmArquivo()` de um objeto da classe **Relatório** recebe um objeto da classe **Filtro** como argumento. No corpo do método `salvarEmArquivo()` é realizada uma chamada ao método `exportar()` do objeto recebido como argumento. Neste caso, podemos dizer que o **Relatório** usa **Filtro**.

Modelagem de Dados

Diagrama de Classe | Dependência | Associação | Generalização

Os relacionamentos de associação vão um pouco mais fundo do que os relacionamentos de dependência.

As associações são relacionamentos estruturais. Uma associação indica que um objeto contém ou que está conectado a outro objeto. (o relacionamento “tem um”)

Quando modelar associações?

Pode-se modelar uma associação quando um objeto usa outro. Uma associação permite que você modele “quem faz o que” em um relacionamento.

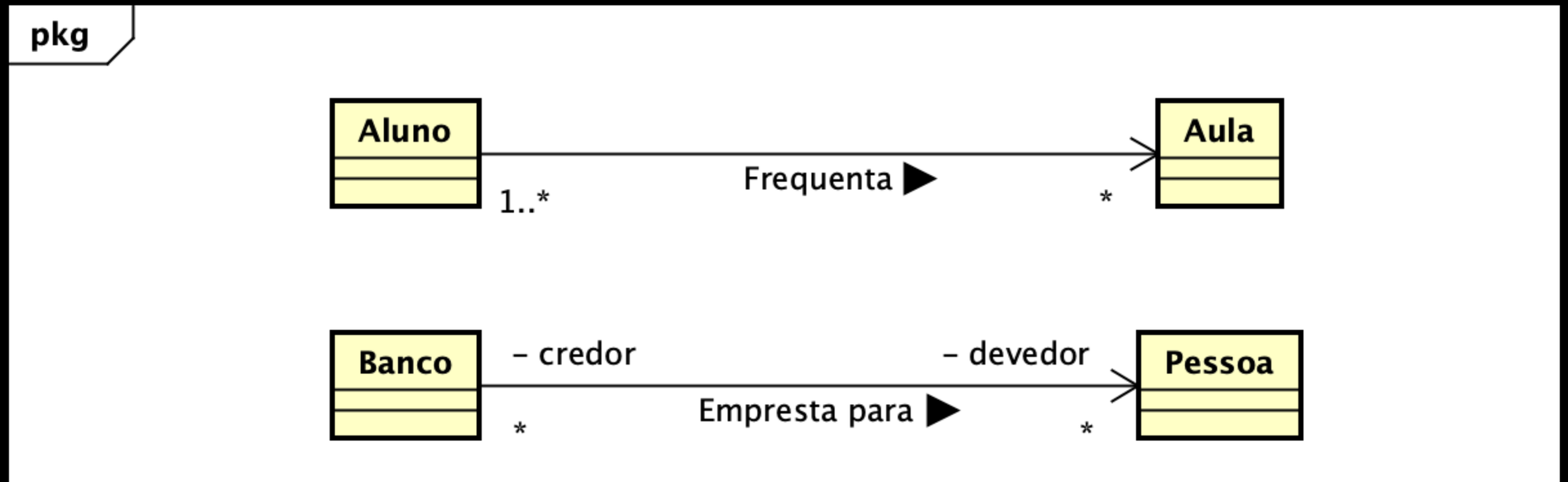
A UML define dois tipos de associação: **agregação** e **composição**. Esses subtipos de associação o ajudam a refinar mais seus modelos.

Modelagem de Dados

Diagrama de Classe | Dependência | Associação | Generalização

Associação (linha contínua): as duas classes são independentes e podem trabalhar juntas.

A visibilidade destaca qual lado consegue enxergar as extremidades da associação, mas ambas as classes podem estabelecer o relacionamento.



Modelagem de Dados

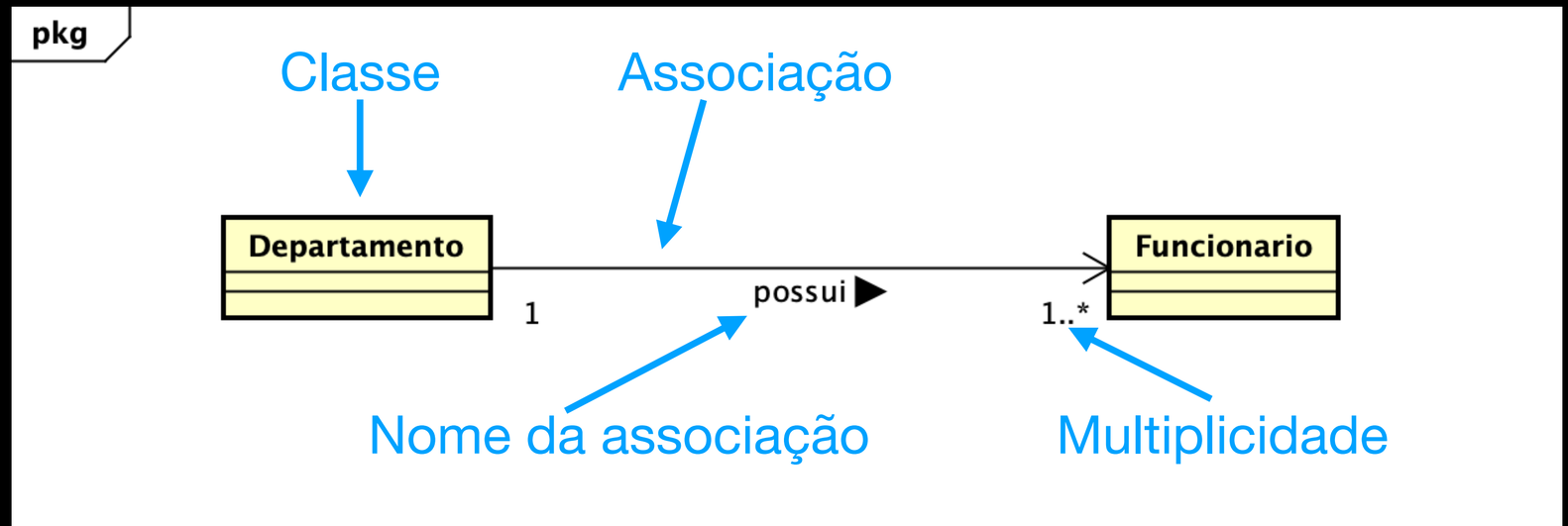
Diagrama de Classe | Dependência | Associação | Generalização

A linha sólida ligando as duas classes representa uma associação entre os objetos das classes

A associação possui um nome (geralmente um verbo) pertencente ao domínio do problema.

Uma seta no final da linha indica que a associação somente pode ser utilizada em uma única direção (associação unidirecional).

Associações possuem multiplicidade, que especifica quantos objetos podem participar da ligação.



Modelagem de Dados

Diagrama de Classe | Dependência | Associação | Generalização
Agregação | Composição

É um tipo especial de associação. Uma agregação modela um relacionamento “tem um” (ou parte de, no jargão da UML) entre pares. Esse relacionamento significa que um objeto não é mais importante do que o outro.

Importância, no contexto de uma agregação, significa que os objetos podem existir independente uns dos outros. Isto é, se o objeto todo deixa de existir, os objetos parte podem continuar existindo.

Exemplo: Equipe e Pessoa.

A **equipe** é quem pode estabelecer o relacionamento, você adiciona pessoas à equipe. Uma **pessoa** por si só não pode “entrar na equipe” . É a equipe que manda no relacionamento, mas mesmo assim. as classes são independentes. Pessoas existem fora de equipes.

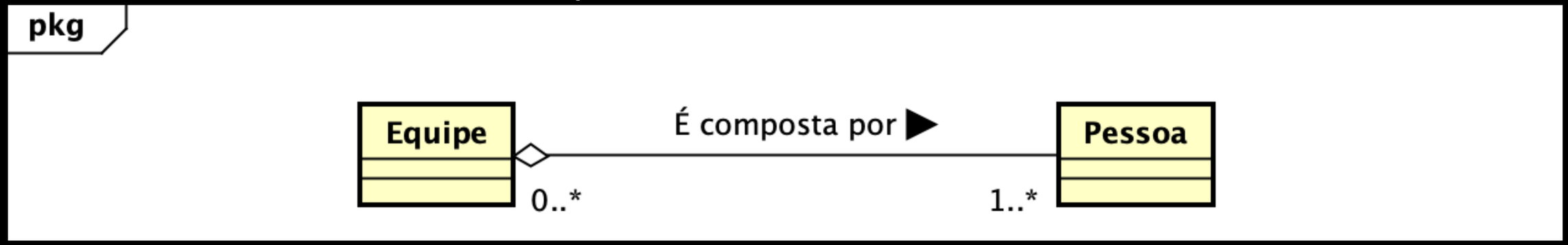
Modelagem de Dados

Diagrama de Classe | Dependência | Associação | Generalização Agregação | Composição

Agregação (linha contínua com diamante branco): a classe que possui o diamante branco controla a associação.

Quando modelar agregação?

Você deve modelar uma agregação quando o objetivo de seu modelo for descrever a estrutura de um relacionamento de pares. Uma agregação mostra explicitamente o relacionamento estrutural todo/parte.



Entretanto, se você estiver mais interessado em modelar “quem faz o que” em um relacionamento, é melhor usar uma associação simples: sem o losango.

Modelagem de Dados

Diagrama de Classe | Dependência | Associação | Generalização Agregação | Composição

A composição é um pouco mais rigorosa do que a agregação. A composição não é um relacionamento entre pares. Os objetos não são independentes uns dos outros. Em vez disso, a parte é dependente do todo. Isso significa, em termos de programação, que quando o objeto todo é destruído, todos os objetos parte são automaticamente destruídos também.

Quando modelar uma composição?

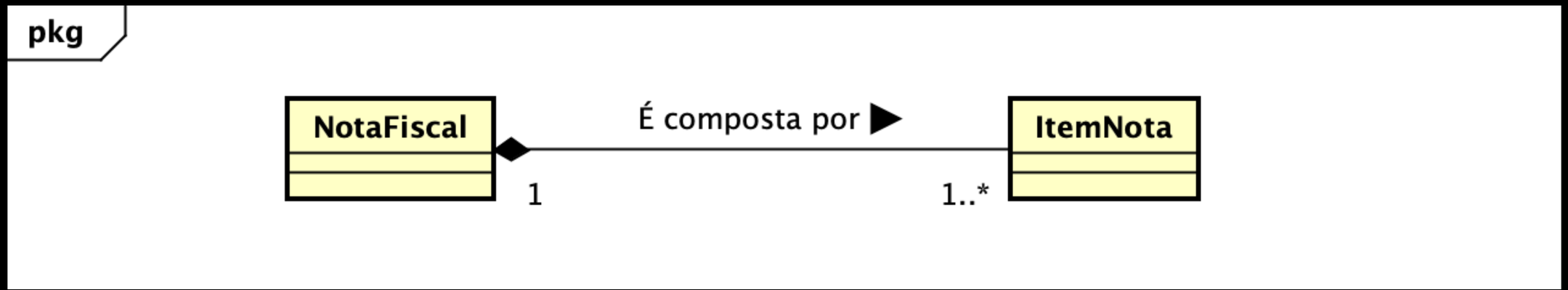
Assim como a agregação, você deve modelar uma composição quando o objetivo de seu modelo for descrever a estrutura de um relacionamento. Uma composição mostra explicitamente o relacionamento estrutural todo/parte.

Modelagem de Dados

Diagrama de Classe | Dependência | Associação | Generalização
Agregação | Composição

Composicão (linha contínua com diamante negro): a composição é uma agregação mais forte.

A classe que possui o diamante controla a associação, e além disso, a outra classe só pode existir associada à classe que tem o diamante e não pode estar associada a outras instâncias.



Modelagem de Dados

Diagrama de Classe | Dependência | Associação | Generalização

Lembre-se de que **agregação** e **composição** são simplesmente refinamentos ou subtipos da associação. Isso significa que você pode modelar agregação e composição como uma associação simples. Tudo depende do que você estiver tentando modelar em seu diagrama.

Agregação: relacionamento “contém-um”. A implementação usa a definição de uma classe externa.

Composição: relacionamento “possui-um”. A implementação pode usar a definição de uma classe interna.

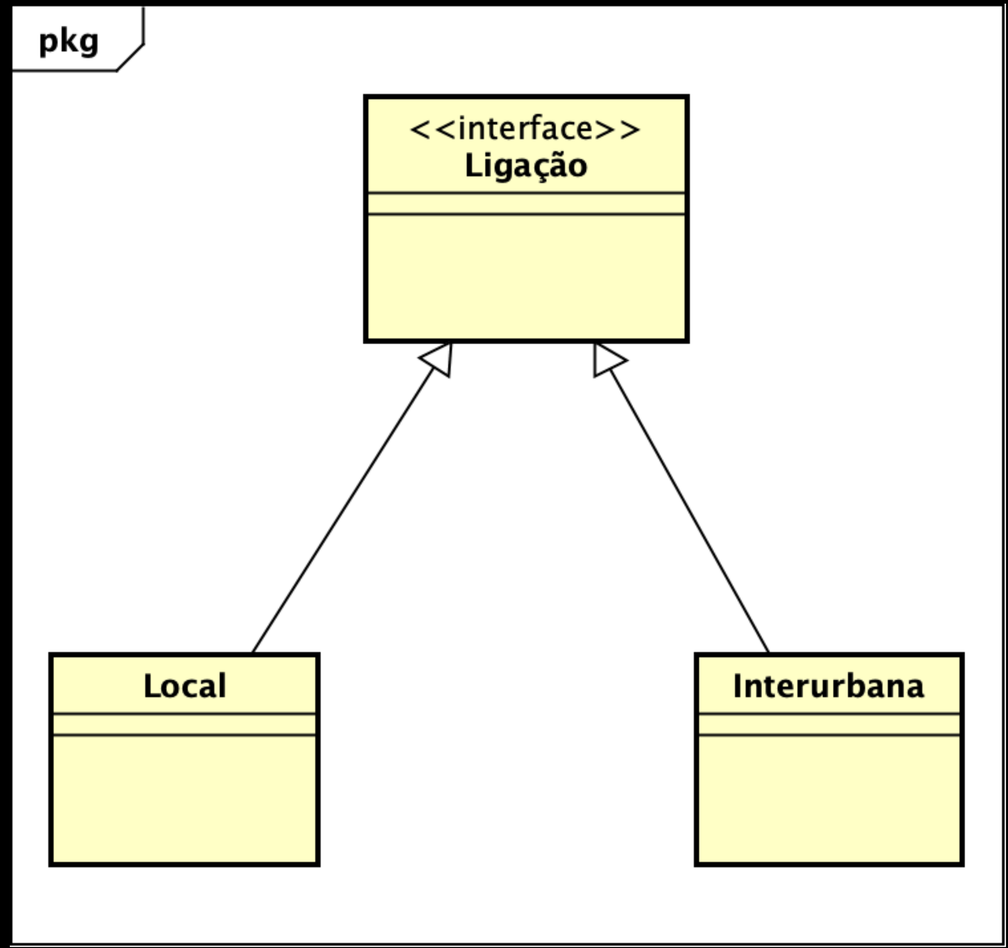
Modelagem de Dados

Diagrama de Classe | Reflexiva

Associação de uma classe com ela mesma.

A classe possui um estereótipo que a identifica como abstrata: abstract.

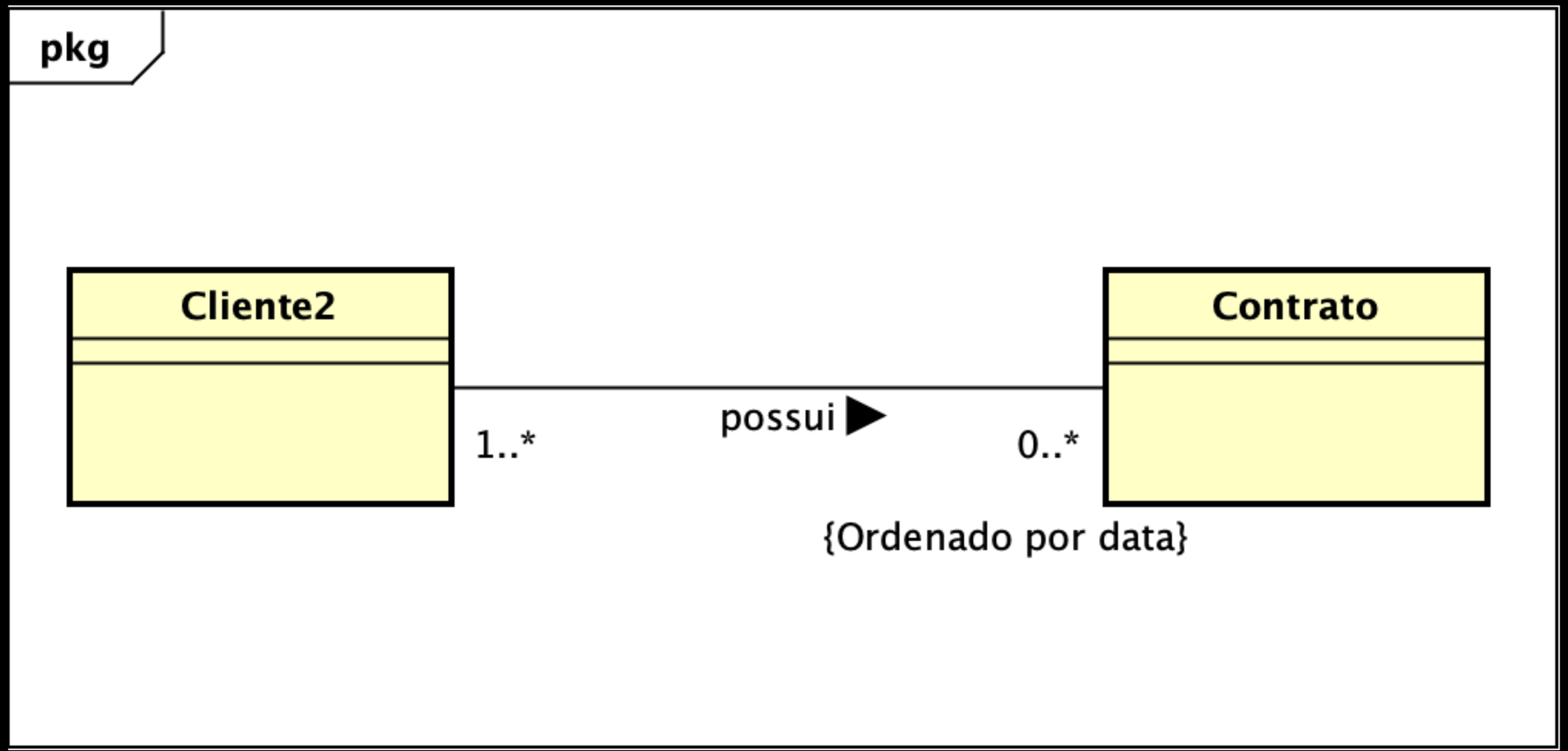
Os relacionamentos com a interface são de herança e realização.



Modelagem de Dados

Diagrama de Classe | Ordenada

Ligação entre os objetos que necessitam assumir uma determinada ordem.



Modelagem de Dados

Diagrama de Classe | Dependência | Associação | Generalização (Herança)

Um relacionamento de generalização é um relacionamento entre o geral e o específico. É a herança.

Se você tem um relacionamento de generalização, então sabe que pode substituir uma classe filha, pela classe progenitora.

A generalização incorpora o relacionamento “é um”, que permite que você defina relacionamentos com capacidade de substituição.

É o relacionamento entre um elemento mais geral e um elemento mais específico (respectivamente, superclasse e subclasse).

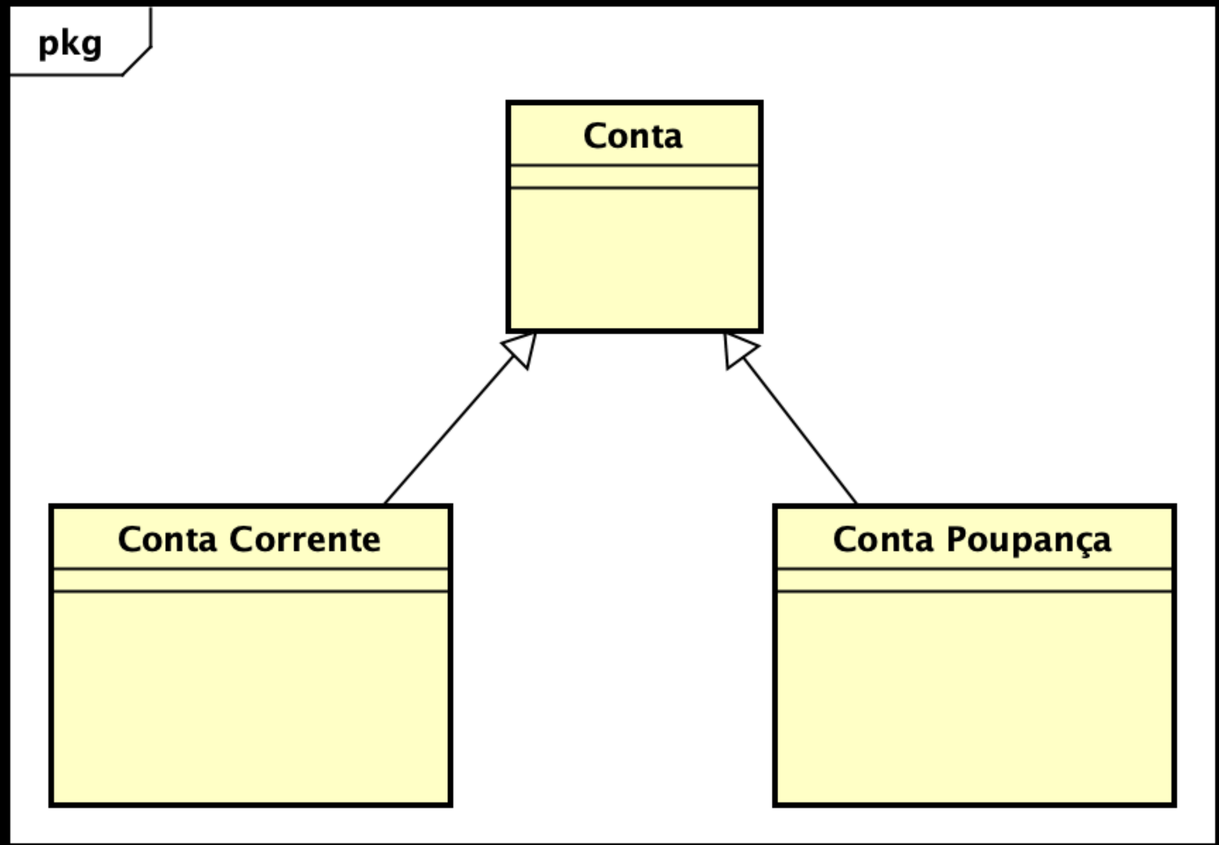
O elemento mais específico pode conter somente informação adicional acerca do elemento mais geral.

Modelagem de Dados

Diagrama de Classe | Dependência | Associação | Generalização (Herança)

Através de **relacionamentos com capacidade de substituição**, você pode **usar descendentes em vez de seus ancestrais**, ou filhos em vez de seus progenitores.

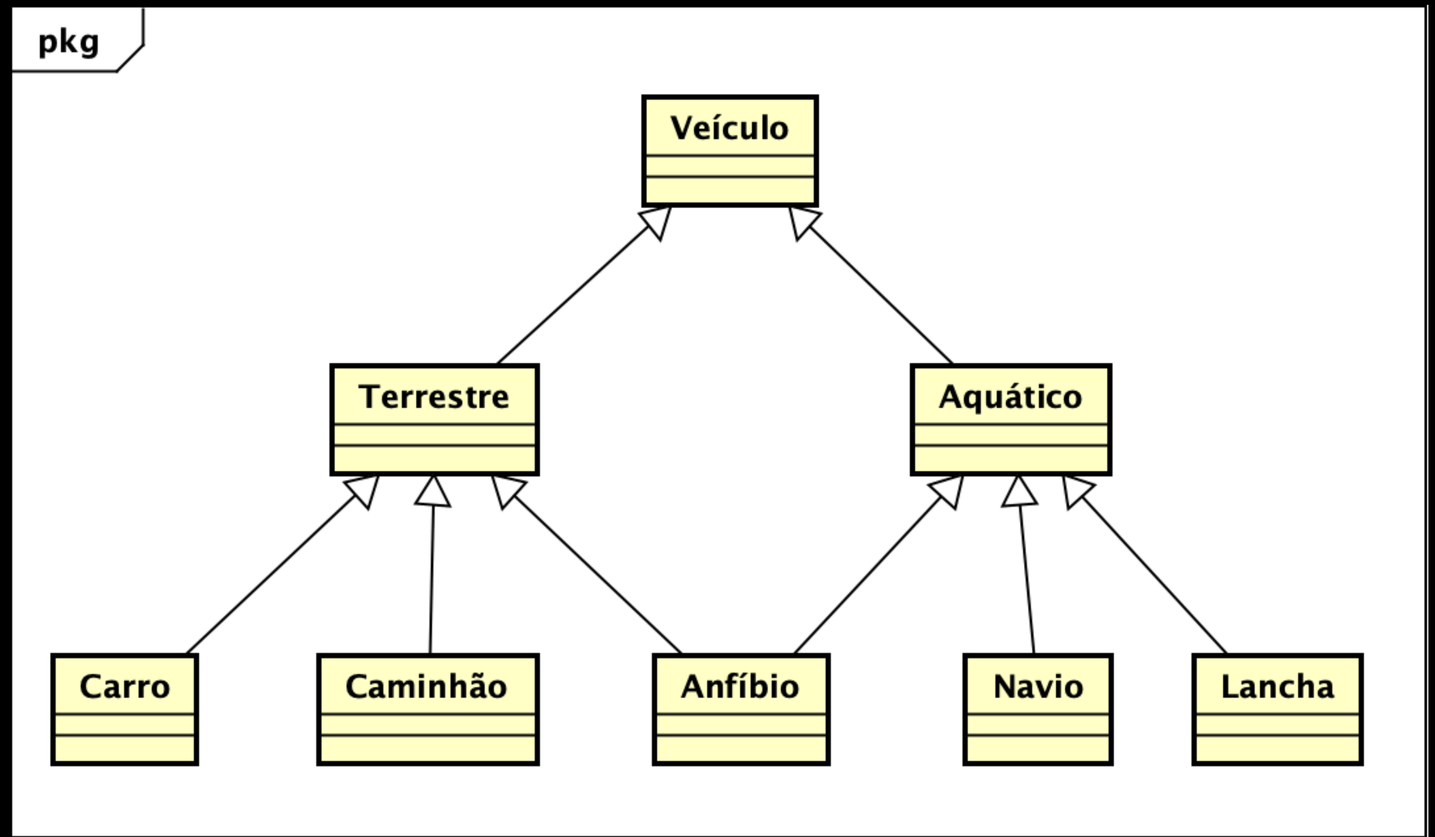
Isto significa, em termos de programação, que se um método, numa linguagem de tipagem forte, espera receber um **argumento** do tipo de uma **classe pai, classes filhas** poderão perfeitamente serem passadas como argumento para o referido **método**. Isto é a **capacidade de substituição**.



Modelagem de Dados

Diagrama de Classe | Dependência | Associação | Generalização (Herança)

Curiosidade: A herança múltipla não é suportada pelo Java. Apesar de ser suportada pelo C++, seu uso não é recomendado.



Modelagem de Dados

Diagrama de Classe