Relacionamentos Entre Classes

O que são?

As classes não existem no vácuo. Em vez disso, elas têm relacionamentos complexos entre si. Esses relacionamentos descrevem como as classes interagem umas com as outras. Os relacionamentos entre as classes também definem responsabilidades. Os relacionamentos possíveis entre as classes são: Dependência, Associação, Agregação, Composição e Herança (Generalização/Especialização).

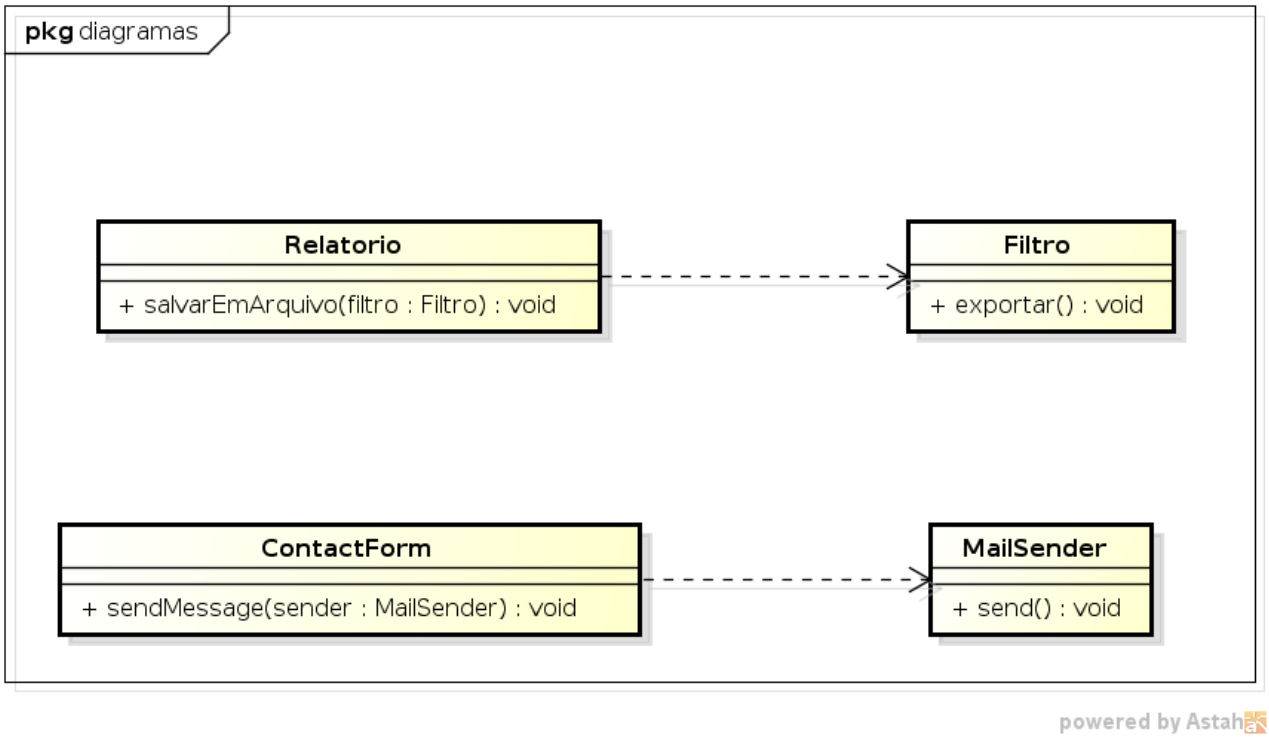
Dependência

Dependência 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).

A lógica de programação pode ser concebida pela mente treinada e pode ser representada em qualquer uma das inúmeras linguagens de programação existentes.

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. Nem todas as dependências são criadas de modo igual. As dependências de interface geralmente estão corretas, enquanto as dependências de implementação quase nunca são aceitáveis.



Quando você deve 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 `salvarComo()` de um objeto da classe Relatorio recebe um objeto da classe Filtro como argumento. No corpo do método `salvarComo()`, é realizada uma chamada ao método exportar() do objeto recebido como argumento. Neste caso, podemos dizer que o Relatorio usa Filtro.

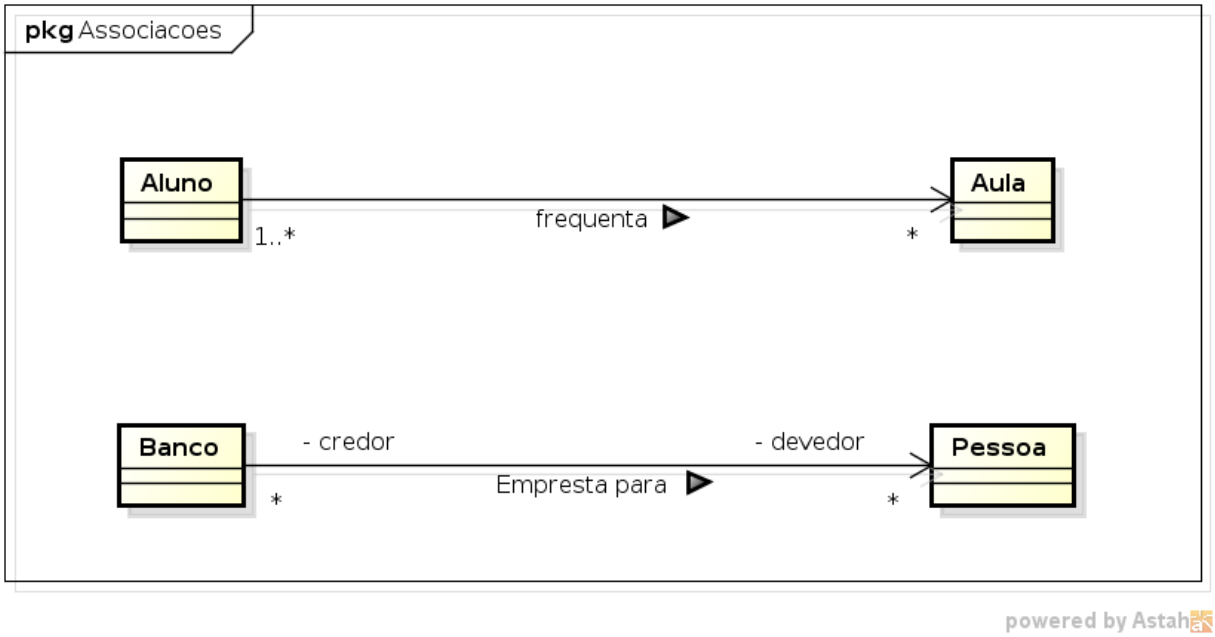
Associação

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.

Como objetos estao conectados, você pode passar de um objeto para outro. Considere a associação entre uma pessoa e um banco, conforme exemplo na figura abaixo.

O nome da associação é um nome que descreve o relacionamento. Cada objeto em uma associação também tem um papel, conforme pode ser visto na figura abaixo.

A multiplicidade indica quantos objetos podem tomar parte em uma associação.



Quando você deve modelar associações?

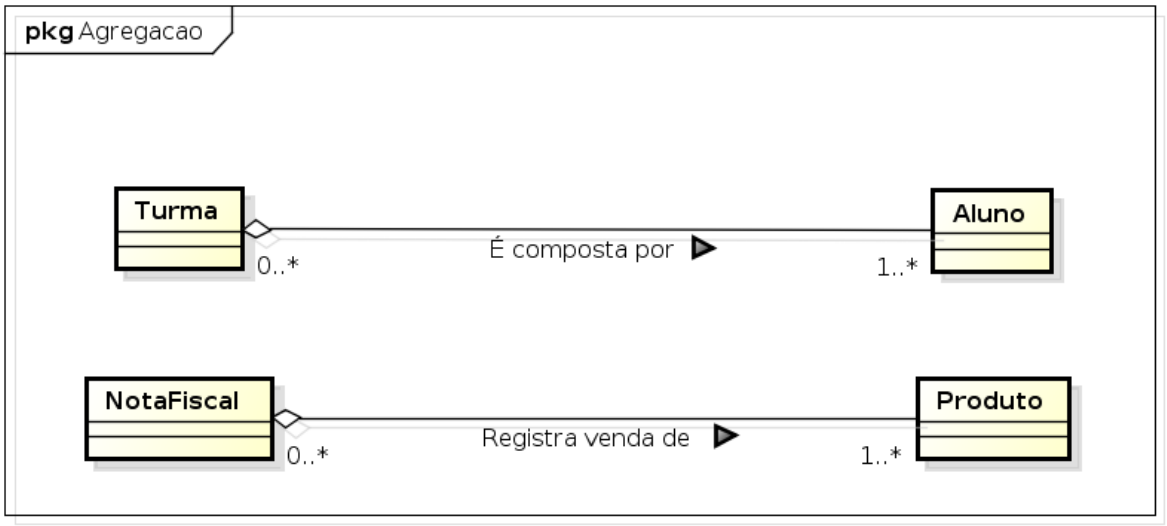
Você deve modelar associações quando um objeto contiver outro objeto - o relacionamento tem um. Você também pode modelar uma associação quando um objeto usa outro. Uma associação permite que você modele quem faz o que em um relacionamento.

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

Agregação

Uma agregação 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 ooutro.

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.



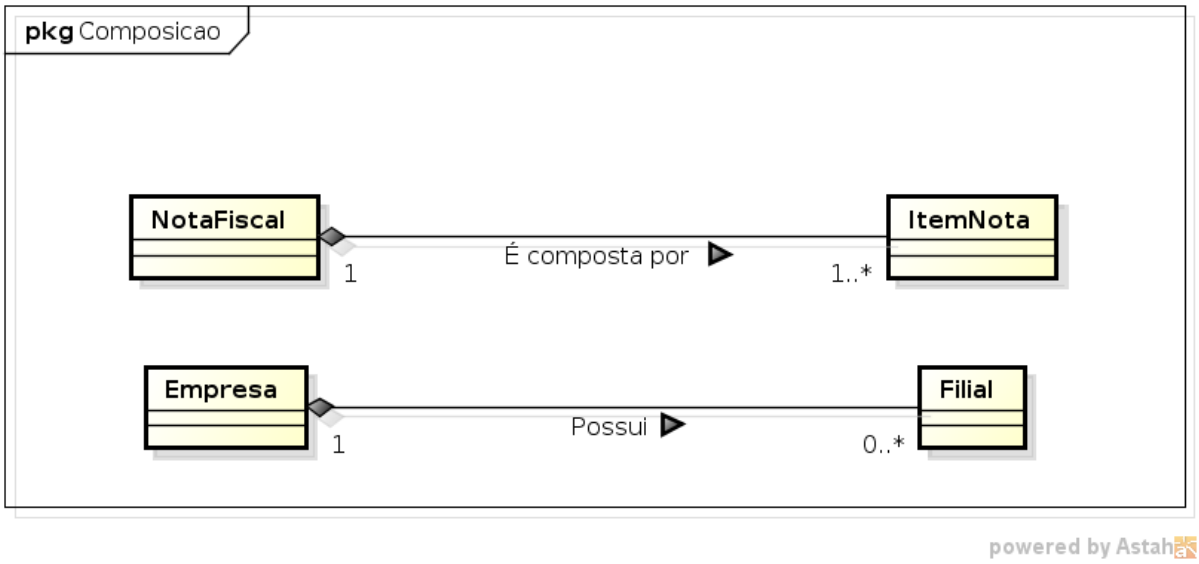
Quando você deve modelar a agregação?

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.

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.



Quando você deve 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.

Ao contrário da agregação, a composição não modela relacionamentos todo/parte entre pares. 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.

Novamente, se o objetivo do seu modelo for capturar os papéis dos objetos na associação, você deve usar uma associção simples.

Nota importante

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.

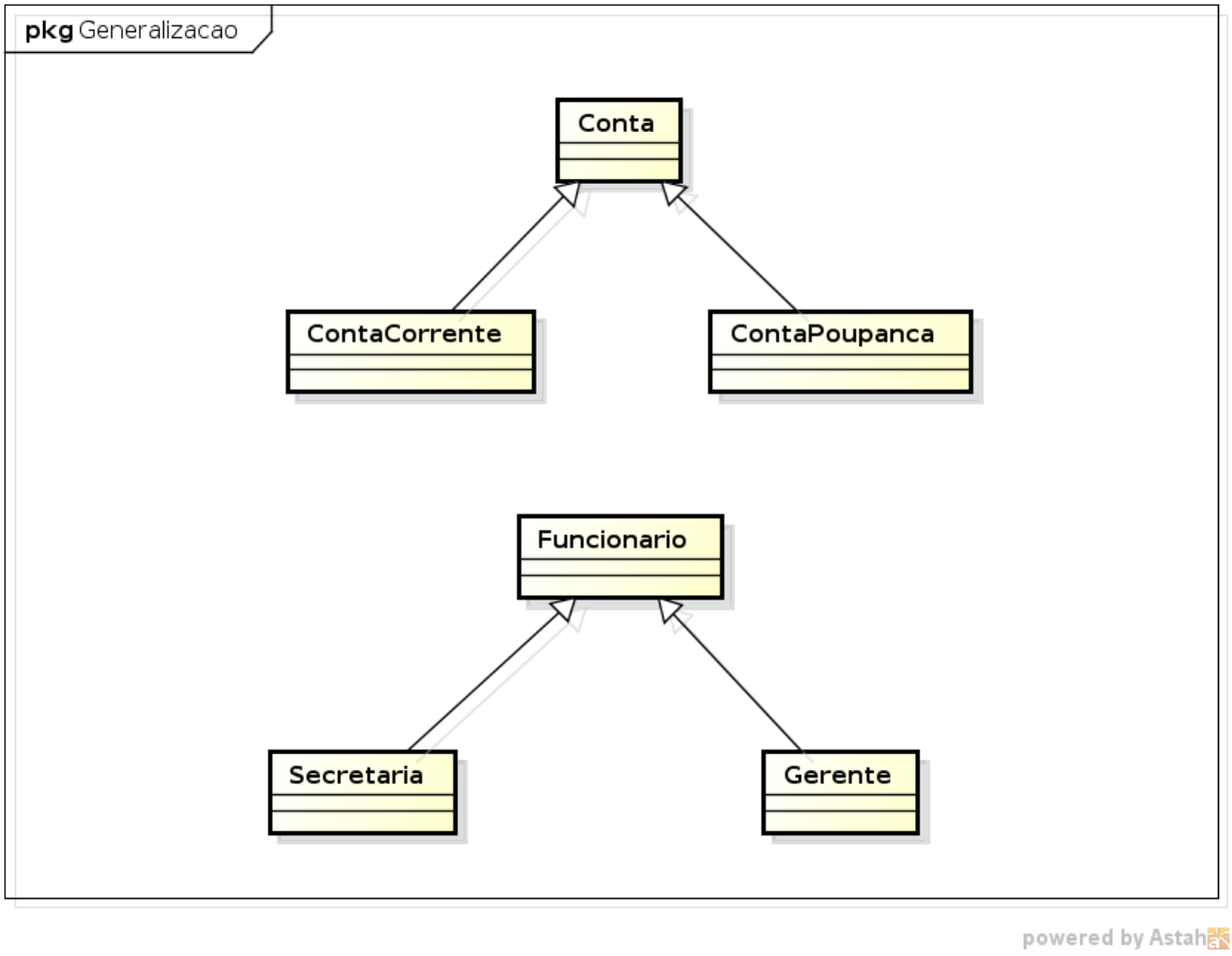
Generalização (Herança)

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.

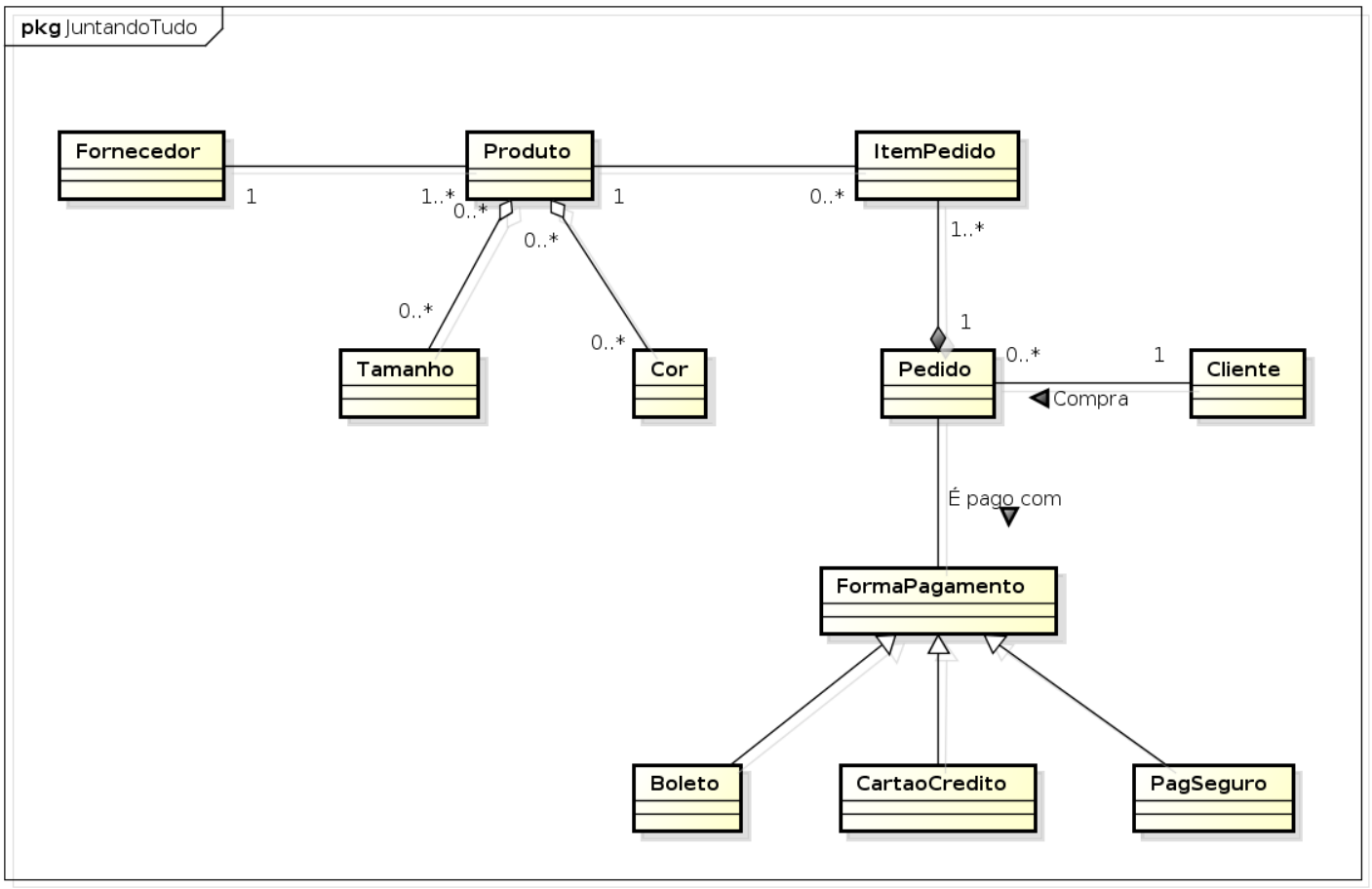
A generalização incorpora o relacionamento *é um*, que permite que você defina relacionamentos com capacidade de substituição.

Através de relacionamentos com capacidade de substituição, você pode usar descendentes em vez de seus ancestrais, ou filhos em vez de seus progrnitores. 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 passadas como argumento para o referido método. Isto é a capacidade de substituição.



Juntando tudo

No diagrama da figura abaixo, podemos ver a combinação dos diversos relacionamentos entre classes aplicados num único modelo.



Para baixar o modelo do Astah Community com os diagramas utilizados neste tutorial, clique aqui.