Engenharia de Software – ESOFT

UML – Diagrama de Classes

Prof. Leonardo Vieira Barcelos

Diagrama de Classes

- Visa permitir a visualização das **classes** que comporão o sistema junto com os respectivos **atributos** e **métodos**, bem como mostrar como as classes se **relacionam**, complementam e transmitem informações entre si.

Diagrama de Classes

- Composto por:
  - **Classes**
  - **Associações** – Relacionamento entre as classes

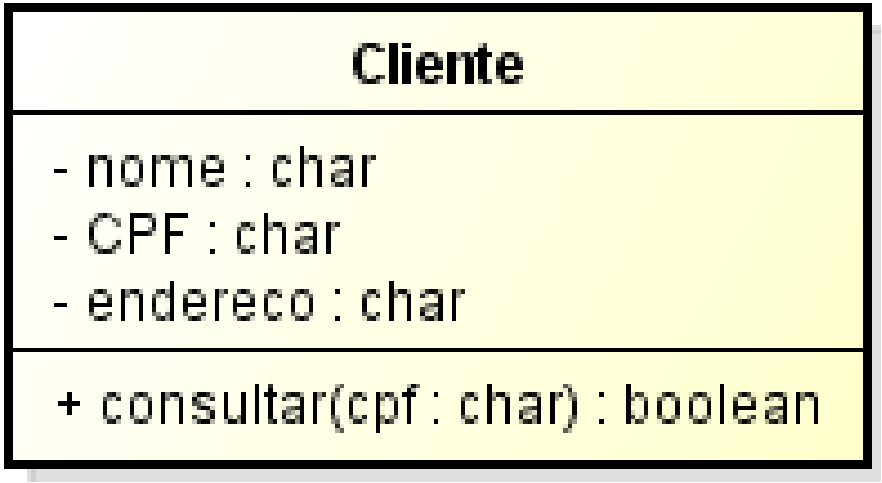
- “Idéia” do diagrama “E-R”

Persistência

- Uma classe é persistente quando é preciso preservar em disco as instâncias da classe.
- Deve ficar claro, que nem toda classe é persistente, não sendo muitas vezes necessário preservar suas informações.

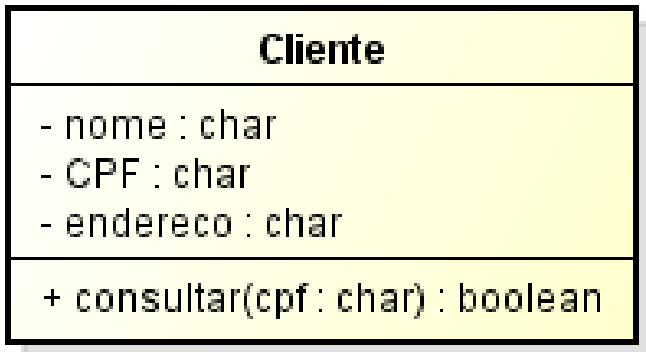
Classes, Atributos e Métodos

- **Atributos** – Armazenam os “dados” dos objetos
- **Métodos** – Funções que uma instância da classe pode executar



Classes, Atributos e Métodos

- Visibilidade
  - “+” = Visibilidade pública – visível em qualquer classe de qualquer pacote.
  - “#” = Visibilidade protegida – visível para classes do mesmo pacote
  - “-” = Visibilidade privada – visível somente para classe

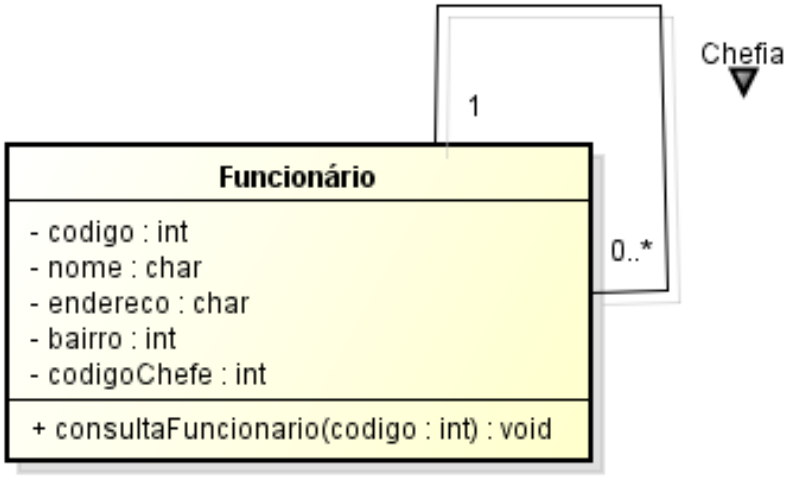


Relacionamentos

- As classes costumam ter relacionamentos entre si com o intuito de compartilhar informações e colaborarem umas com as outras para permitir a execução dos processos.

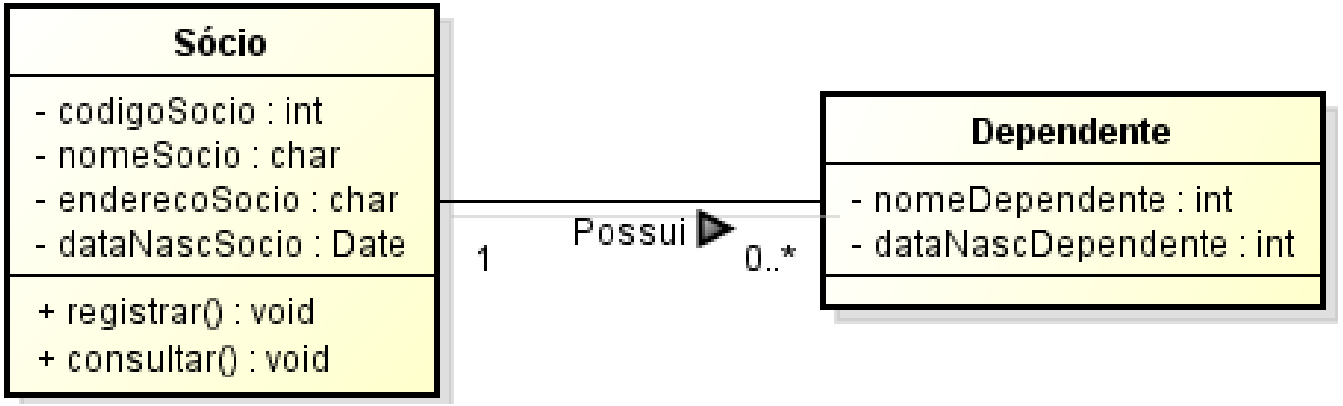
- Os mais comuns são:
  - Associações
  - Especialização/Generalização

Associação Unária ou Reflexiva



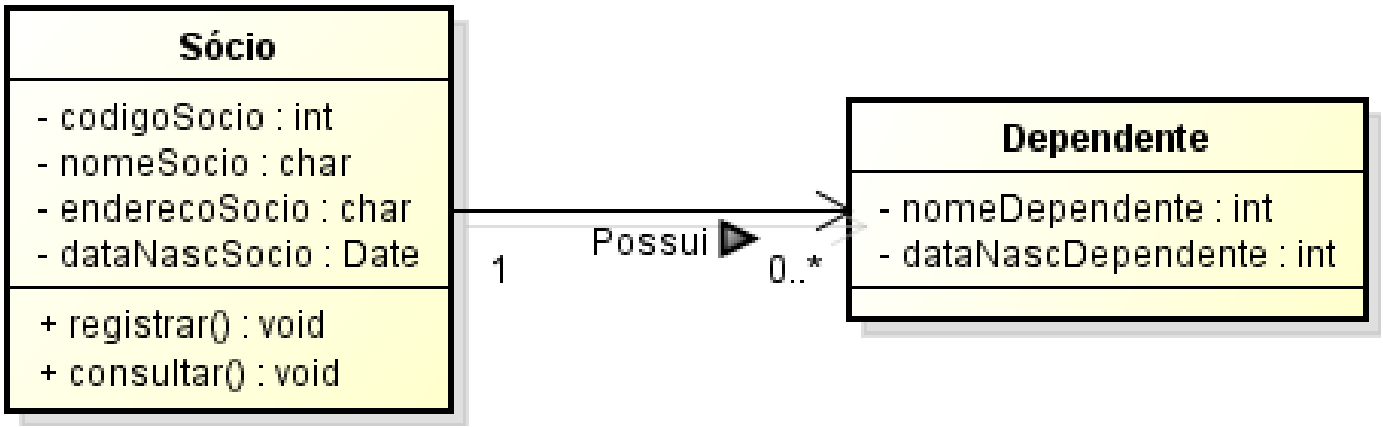
- No exemplo mostrado, a classe Funcionário possui os atributos *Código do funcionário, nome, endereço, bairro e Código do Chefe*.
- O chefe do funcionário também é, por sua vez, um funcionário da empresa e, portanto também se constitui em uma instância da classe Funcionário.
- Esta associação determina que um funcionário pode ou não chefiar outros funcionários, conforme multiplicidade “0..*”.

Associação Binária



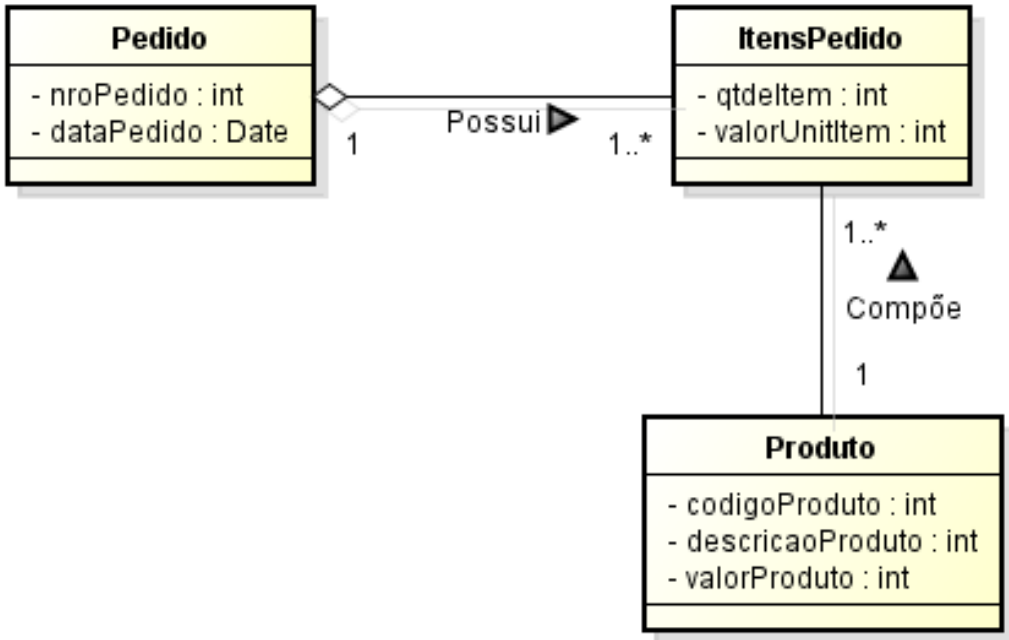
- No exemplo acima, demonstra que um objeto da classe sócio pode relacionar-se ou não com instâncias da classe Dependentes, conforme mostra a multiplicidade 0..*, enquanto que, se existe um objeto da classe Dependente, ele terá que se relacionar obrigatoriamente com um objeto da classe Sócio.

Associação Binária



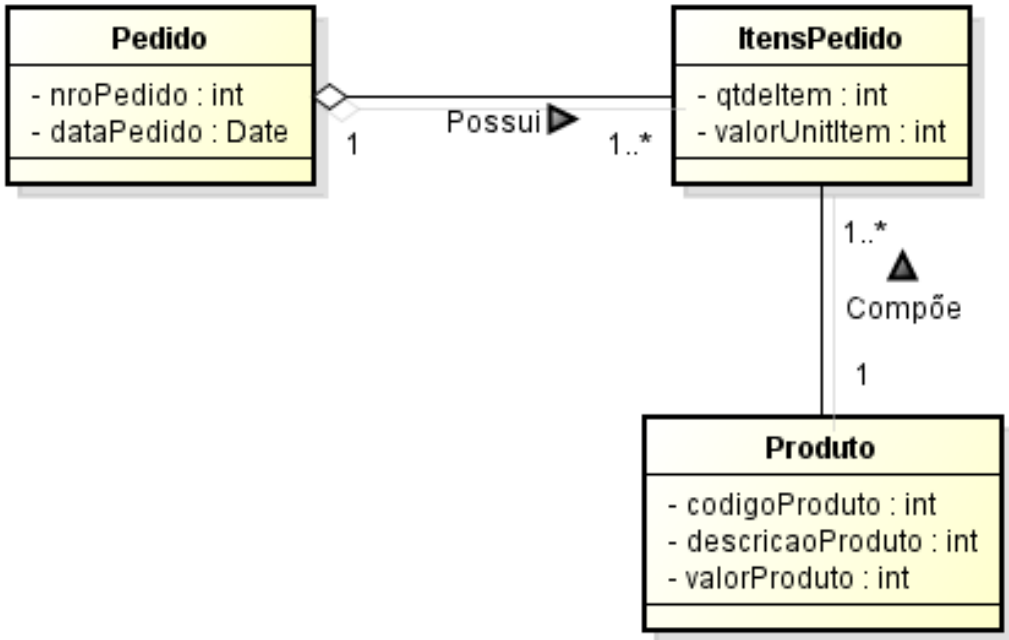
Neste exemplo a associação demonstra a navegabilidade dos dados.

Agregação



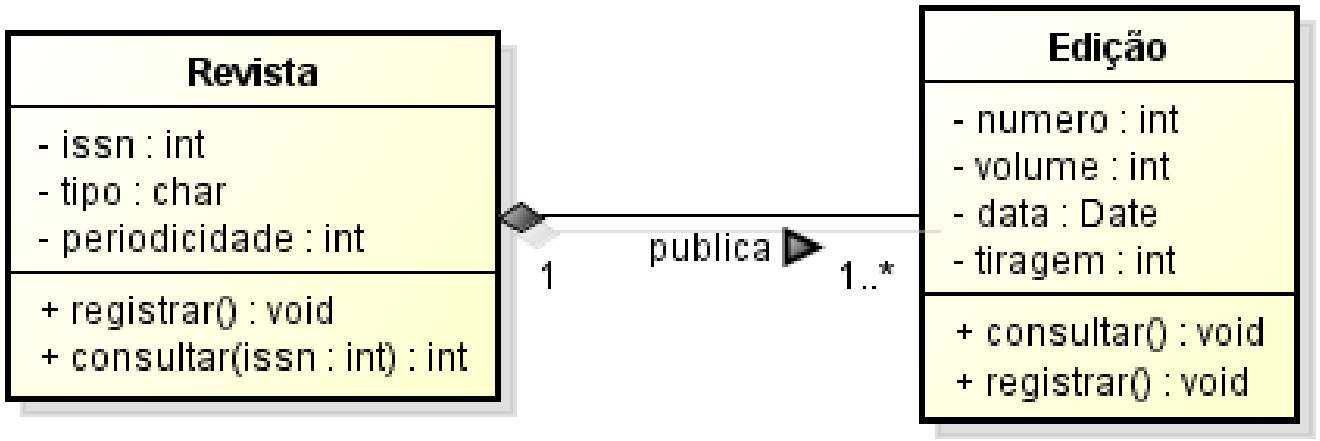
- Agregação é um tipo especial de associação onde tenta-se demonstrar que as informações de um objeto (chamado objeto-todo) precisa ser complementadas pelas informações contidas em um ou mais objetos de outra classe (chamados objeto-parte).
- Neste exemplo, a agregação existe apenas entre a classe Pedido e a classe Itens Pedido. A classe Produto possui apenas uma associação binária comum com a classe Itens Pedido. Podemos perceber isso por meio da multiplicidade 1..* na associação Compõe, na extremidade da Classe Itens Pedido, o que significa que um objeto da classe Produto pode se referir a muitos objetos da classe Itens Pedidos, no entanto um objeto da Classe Itens Pedido refere-se a somente uma instância da classe Produto, já que a multiplicidade na extremidade desta classe é 1..1.

Agregação



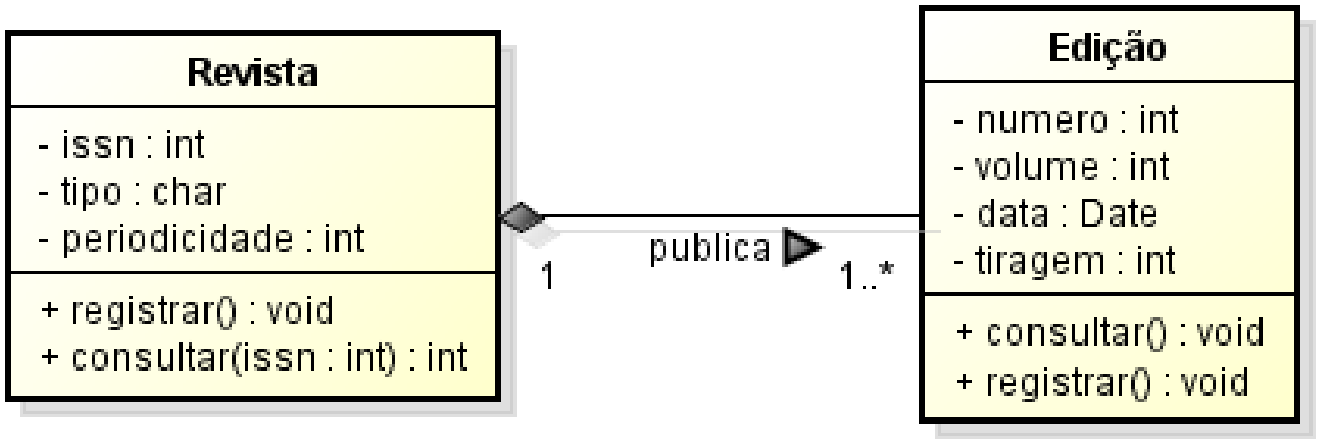
- Já no caso da agregação entre a classe Pedido e a classe Itens Pedido é apresentado uma diferença em relação à associação binária entre a classe Produto e a classe Itens Pedido. Um pedido pode tanto possuir apenas um item como vários e é muito difícil determinar o número máximo de itens que um pedido possa ter.
- No exemplo, as informações da classe Pedido estão incompletas, possuindo apenas atributos que não se repetem, como por exemplo o número de pedido e a data. Os atributos que podem se repetir, no caso referentes aos dados das instâncias da classe Itens Pedido, como por exemplo, a quantidade do item solicitado e seu valor unitário, devem ser armazenados em uma classe dependente da classe Pedido.

Composição



- Uma associação do tipo Composição constitui-se em uma variação da associação de agregação. Uma associação de composição tenta representar um vínculo mais forte entre os objetos-todo e objetos-parte, procurando demonstrar que os objetos-parte têm de pertencer exclusivamente a um único objeto-todo com que se relacionam.
- Em uma composição, um mesmo objeto-parte não pode associar-se a mais de um objeto todo e não podem existir sem o objeto-todo.

Composição

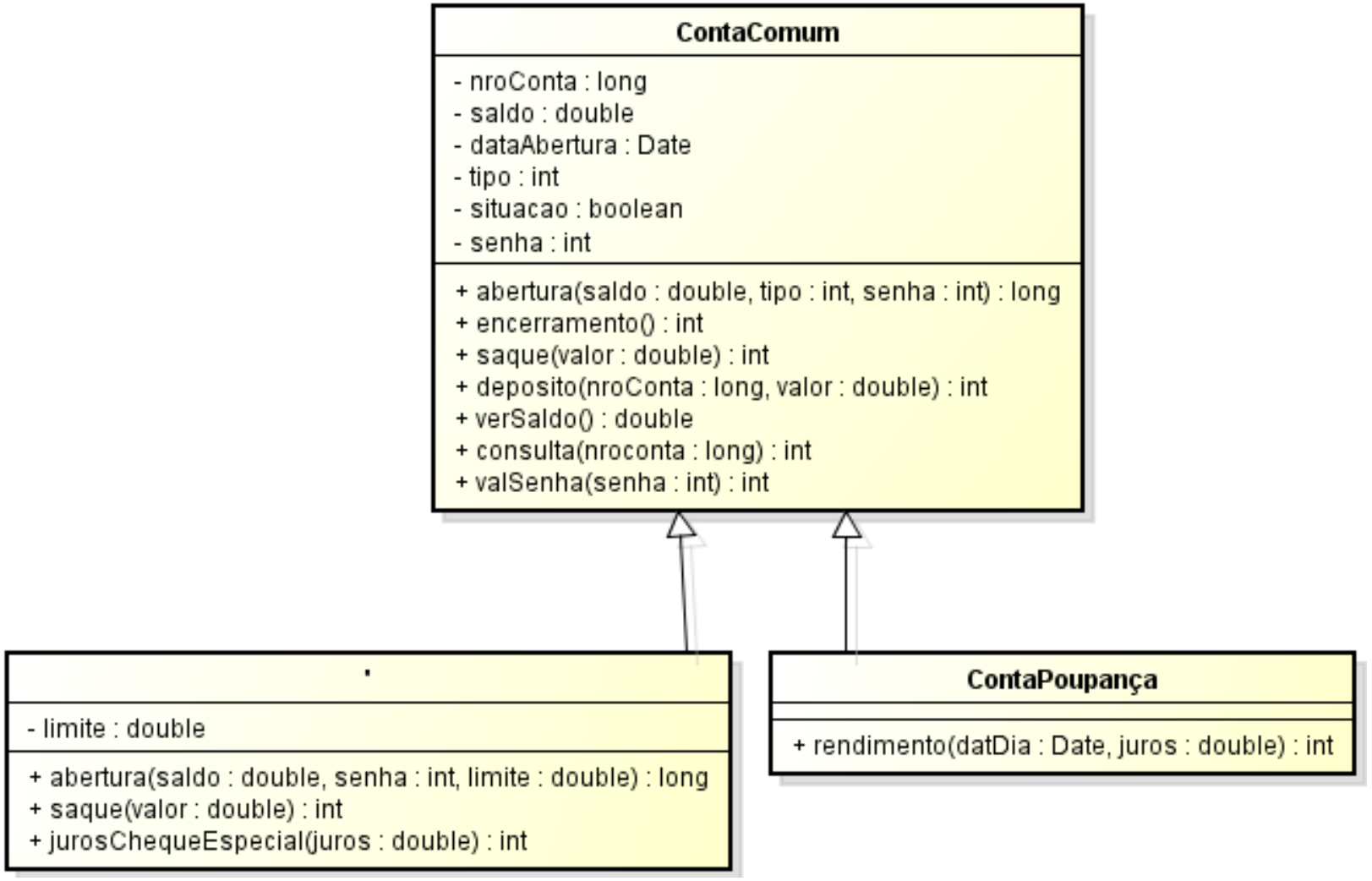


- Observando a figura percebe-se que um objeto da classe Revista refere-se à no mínimo um objeto da classe Edição, podendo referir-se a muitos objetos desta classe, e que cada instância da classe Edição relaciona-se única e exclusivamente a uma instância da classe Revista Científica, não podendo relacionar-se com nenhuma outra.

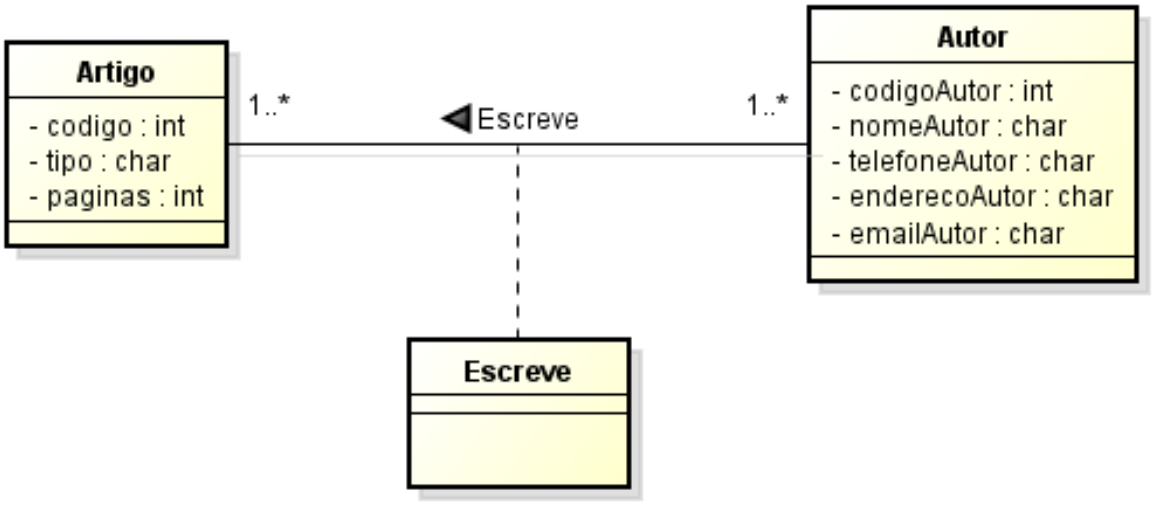
Especialização/Generalização

- Seu objetivo é identificar classes-mãe, chamadas gerais e classes-filhas, chamadas especializadas. Este tipo de relacionamento permite também demonstrar a ocorrência de métodos polimórficos nas classes especializadas do sistema.
- A especialização/generalização ocorre quando existem duas ou mais classes com características semelhantes, assim para evitar ter que declarar atributos e/ou métodos idênticos e como uma forma de reaproveitar código, cria-se uma classe geral onde são declarados os atributos e métodos comuns a todas as classes envolvidas no processo e então declaram-se classes especializadas ligadas à classe geral, que herdam todas as suas características podendo possui atributos e métodos próprios.

Especialização/Generalização

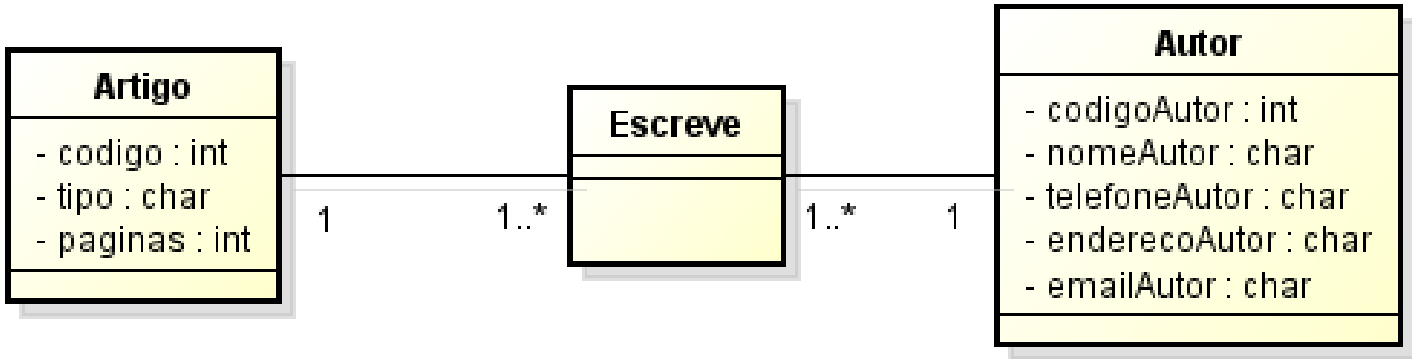


Classe Associativa



- Classes associativas são classes produzidas quando da ocorrência de associações que possuem multiplicidade muitos (*) em todas as suas extremidades. As classes associativas são necessárias nesses casos porque não existe um repositório que possa armazenar as informações produzidas pelas associações já que todas as classes apresentam multiplicidade muitos.

Classe Associativa



- Classes associativas podem perfeitamente ser substituídas por classes normais, chamadas neste caso Classes Intermediárias, mas que desempenham exatamente a mesma função das classes associativas.