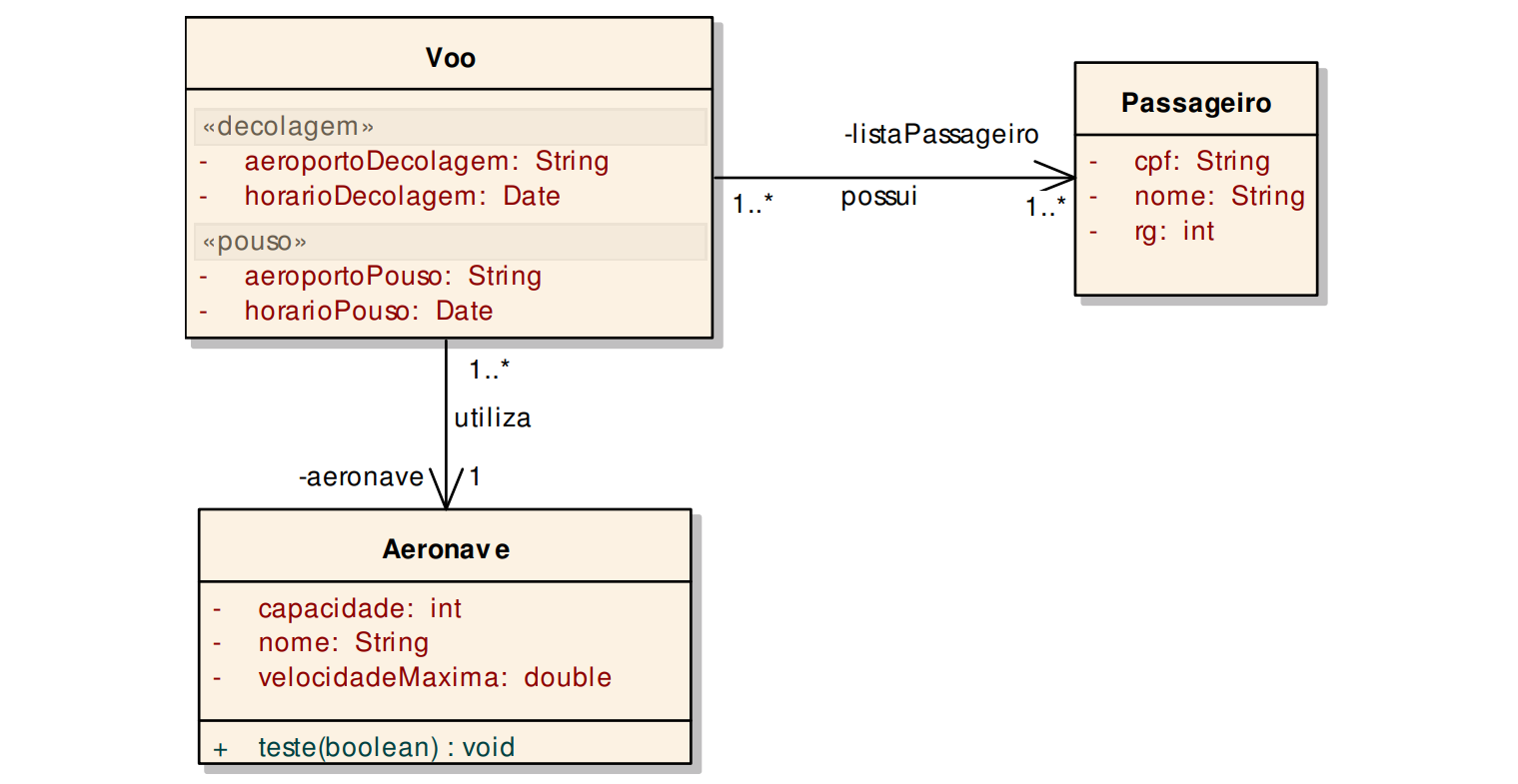
**INTRODUÇÃO**

Segundo seus próprios criadores Booch, Rumbaugh e Jacobson (2005), a UML (Unified Modeling Language) é uma linguagem voltada à visualização, especificação, construção e documentação de sistemas a partir de representações conceituais do Software em questão. Um dos principais diagramas da UML é o Diagrama de Classes, tema do nosso trabalho:

**No Slide:** Um diagrama de classes mostra a estrutura estática do modelo, onde seus elementos são representados por classes, explicitando sua estrutura interna e seus relacionamentos com demais classes (LIMA, 2005).

**Fala:** Um diagrama de classe usa representações visuais de classes para descrever a estrutura estática que devem estar presente no sistema a ser modelado; em outras palavras, a estrutura interna de um objeto é especificada quanto aos seus atributos, métodos e relacionamentos, e organizada em uma classe visual.

**ESTRUTURA DA CLASSE**



**Fala:** As classes são representadas visualmente por um retângulo separado em três partes:

A parte superior contém o nome da classe, que deve começar com letra maiúscula e ser único. A parte do meio contém os atributos, que são as características do objeto e carregam suas informações, e a parte inferior os métodos ou operações, que são os comportamentos ou ações.

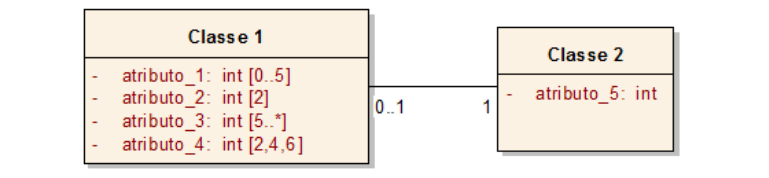
**ATRIBUTOS:** [visibilidade] [/] nome [: tipo] [multiplicidade] [= valor padrão]

**MÉTODOS:** [visibilidade] nome ([lista de parâmetros]): [tipo do retorno]

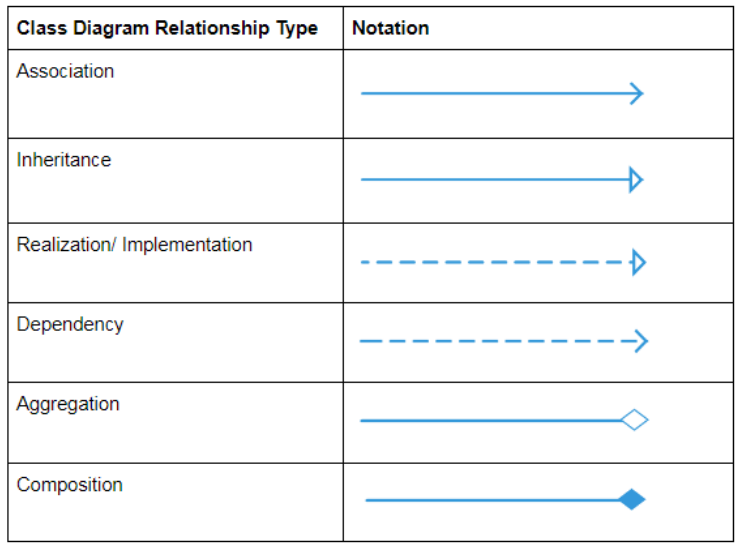
Observe que os atributos e os métodos possuem visibilidade especificadas por modificadores de acesso que limitam seu escopo. Os mais comuns são: Público (+), Privado (-), Protegido (#). Quando publico o acesso é geral; privado o acesso ocorre apenas dentro da classe; protegido, permite que os métodos e atributos sejam acessados pelas classes filhas.

Observe também que é necessário especificar o tipo do atributo. Esse tipo pode ser um dado primitivo, mas também uma referência a outra classe. Dessa forma, é possível estabelecer vínculo com outras classes. Quando isso é feito por meio de listas, elas podem ser delimitadas pela multiplicidade, que é a quantidade de instancias que podem estar associadas. Texto, Tabela

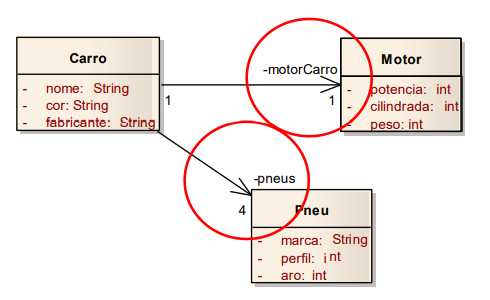
Descrição gerada automaticamente



**ASSOCIAÇÃO**

****

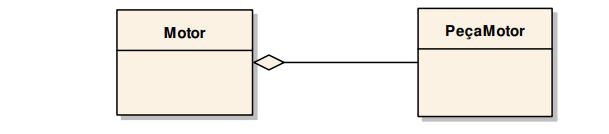
**ASSOCIAÇÃO SIMPLES:** relacionamento entre classes totalmente independentes.



Um carro possui 1 motor; um carro possui 4 pneus.

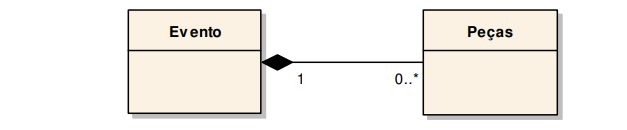
Observe a navegabilidade, essa associação é direcional, o que significa que Pneu e Motor não conhecem Carro. Note também que Motor e Pneu existem independentemente de Carro

**AGREGAÇÃO:** um objeto é parte de outro, mas a parte pode existir sem o todo



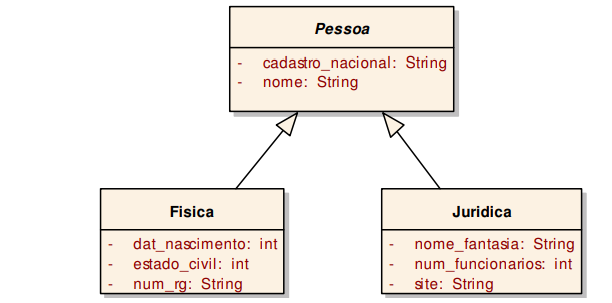
Motor e PeçaMotor existe paralelamente, mas o Motor apenas funcionará corretamente se conter todas as peças necessárias.

**COMPOSIÇÃO:** o objeto não existe sem a relação



Imagine este Evento sendo um espetáculo teatral, o show só pode ocorrer se as Peças forem executadas. Sem vínculo com as Peças, não existe o Evento, bem como, não tem sentido haver peças se não há eventos para apresentá-las.

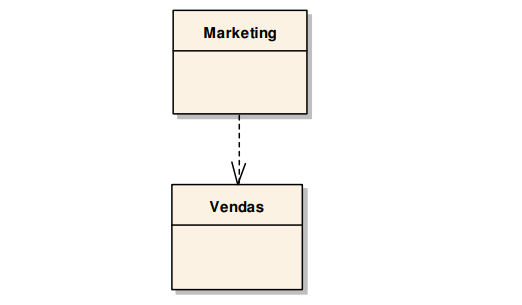
**HERANÇA / GENERALIZAÇÃO:** uma classe implementa ou estende outra de forma a adquirir tudo que a classe herdada possui.



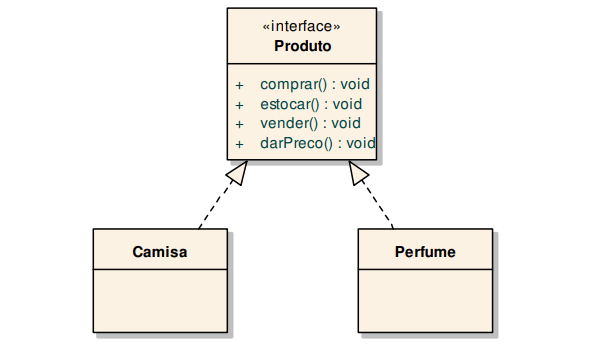
As classes Física e Jurídica possuem seus atributos específicos, mas por estenderem a classe Pessoa, obrigatoriamente herdem os atributos cadastro\_nacional e nome da classe Pessoa. Os métodos e abributos herdados, no entanto, podem ser sobrecarregados ou sobrescritos.

Note também nesse exemplo que a classe Pessoa está com seu nome grafado em itálico, isso indica que é uma classe abstrata. Uma classe abstrata não pode ser instanciada, assumindo apenas o papel de superclasse.

**DEPENDENCIA:** uma classe precisa acessar recursos específicos de outra classe, e em caso de mudança na classe fornecedora, é necessário mudar também na classe que utiliza.

****

**INTERFACE: uma interface não possui atributos, e seus métodos são apenas definidos. Consiste na estrita assinatura de o que precisa ser implementado pela classe que a implementar.**

****

Referência

<https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-classe-uml>

https://creately.com/blog/pt/diagrama/tutorial-diagrama-de-classes/

https://www.significados.com.br/diagrama-de-classes/