=======================================================

***Diagrama de Classes***

=======================================================

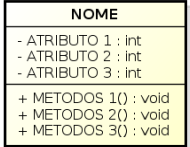
* Mostra um conjunto de classes e seus relacionamentos
* É o diagrama central da modelagem orientada a objetos
* Demonstra toda a estrutura do software

=======================================================

***Elementos de um Diagrama de Classes***

=======================================================

* Graficamente, as classes são representadas por retângulos incluindo nome, atributos e métodos.



* Devem receber o nome de acordo com o vocabulário do problema
* É comum adotar um padrão para nomeá-las

– Ex: todos os nomes de classes serão substantivos singulares com a primeira letra maiúscula.

# ***Atributos:***

– Representam o conjunto de características dos objetos daquela classe.

– **Visibilidade:**

**+ Público:** Visível para qualquer classe de qualquer pacote

**# Protegido:** Visível para classe do mesmo pacote

***-* Privado:** Visível somente para a classe

– **Exemplo:**

**+ nome** : String

**# nome** : String

**-** **nome** : String

# ***Métodos:***

– Representam o conjunto de operações(comportamento) que a classe fornece.

– Visibilidade:

**+ Público:** Visível para qualquer classe de qualquer pacote

**# Protegido:** Visível para classe do mesmo pacote

***-* Privado:** Visível somente para a classe

– **Exemplo:**

+ getNome() : String

# setNome(String nome) :void

- getNome() : String

# ***Relacionamentos:***

Os relacionamentos possuem:

– **Nome:** Descrição dada ao relacionamento (faz, tem, possui, ...)

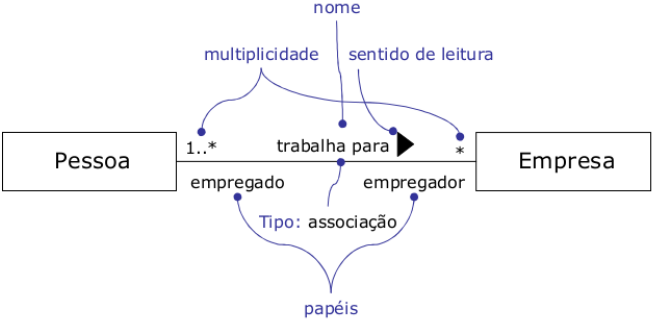
– **Sentido de leitura**

– **Navegabilidade:** indicada por uma seta no fim do relacionamento

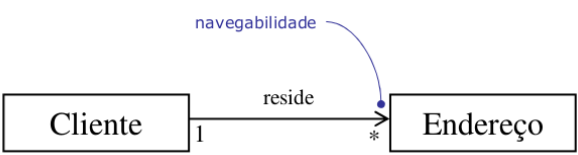
– **Multiplicidade:** 0..1, 0..\*, 1, 1..\*

– **Tipo:** Associação (agregação, composição), generalização e dependência

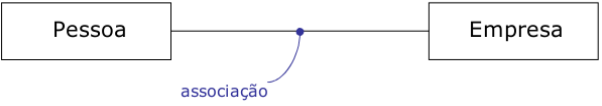
– **Papéis:** Desempenhado por classes de um relacionamento



* O cliente sabe quais são seus endereços, mas o endereço não sabe quais são seus clientes.



* Uma associação é um relacionamento estrutural quem indica que os objetos de uma classe estão vinculados a objetos de outra classe.
* Uma associação é representada por uma linha sólida conectando as duas classes.



* **Indicadores de multiplicidade:**

– **1** Exatamente um

– **1..\*** Um ou muitos

– **0..\*** Zero ou muitos

– **\*** Zero ou muitos

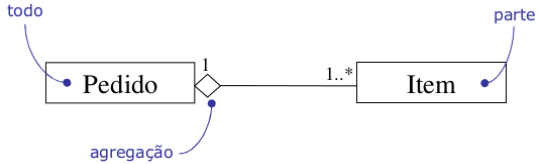
– **0..1** Zero ou um

## ***Agregação:***

É um tipo especial de associação

Utilizada para indicar “Todo-Parte”

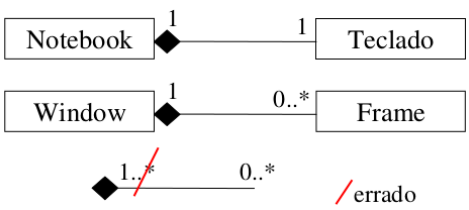
Um objeto “parte” pode fazer parte de vários objetos “todo”



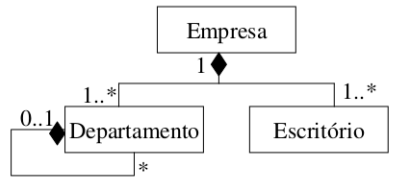
# ***Composição:***

É uma variante semanticamente mais “forte” da agregação

Os objetos “parte” só podem pertencer a um único objeto “todo” e tem o seu tempo de vida coincidente com ele

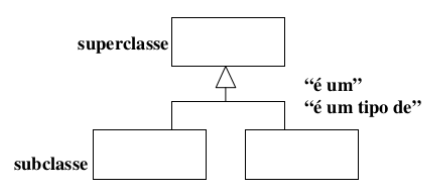
Quando o “todo” morre todas as suas “partes” também morrem

**Exemplo de composição:**



# ***Generalização:***

É um relacionamento entre itens gerais (superclasses) e itens mais específicos (subclasses)



# ***Dependência:***

Representa que a alteração de um objeto (o objeto independente) pode afetar outro objeto (objeto dependente)

**Obs:**

– A classe cliente depende de alguns serviços da classe fornecedor

– A mudança de estado do fornecedor afeta o objeto do cliente

– A classe cliente não declara seus atributos um objeto do tipo fornecedor

– Fornecedor é recebido por parâmetro de método



# ***Classe Associativa:***

Usada quando uma associação entre duas classes contiver atributos da associação

– **Ex:** Professor, Turma, Aula → “horário da aula”

