



INSTITUTO SUPERIOR POLITECNICO DE TECNOLOGIAS E CIENCIAS

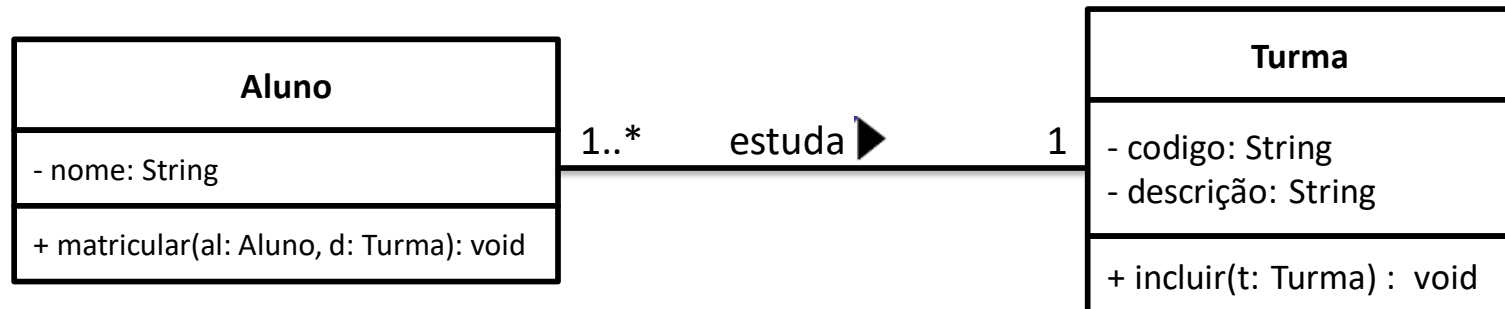
**1º Semestre**

2025/26

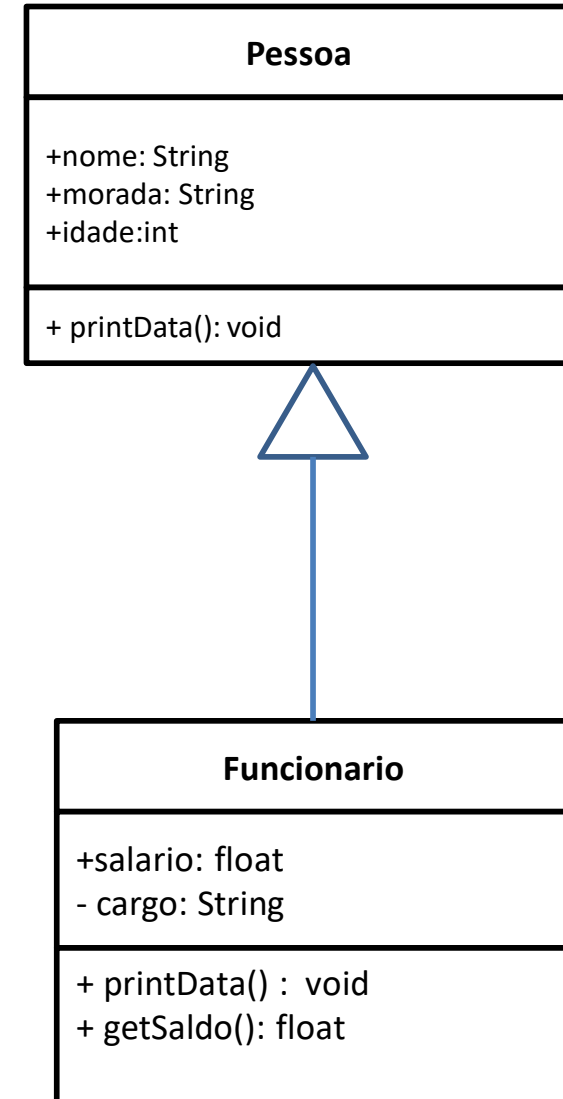


- **Diagrama de Classes**
- Notação UML para classes
- Relacionamentos
  - Associação
    - Agregação
    - Composição
  - Classe associativa
  - Generalização
- Interfaces

- ❖ Diagrama mais utilizado da UML (**Unified Modeling Language**).
- ❖ Permite a visualização das classes utilizadas pelo sistema e como elas se relacionam.
- ❖ Apresenta uma visão estática de como as classes estão organizadas a fim de definir sua estrutura lógica.
- ❖ Foi projectado para ser uma evolução (e não substituição) do Modelo Entidade-Relacionamento da Base de Dados.



- ❖ Uma classe não corresponde, obrigatoriamente, a uma tabela numa base de dados.
  - ❖ Exemplos
    - Classes de Interface
    - Controllers
- ❖ Eventualmente, os atributos de uma classe correspondem aos atributos de uma tabela, mas uma classe não é uma tabela.
- ❖ No modelo lógico de uma base de dados, os métodos de uma classe podem corresponder às operações realizadas sobre uma tabela(criar, alterar, remover, consultar).



➤ Diagrama de Classes

➤ Notação UML para classes

➤ Relacionamentos

➤ Associação

➤ Agregação

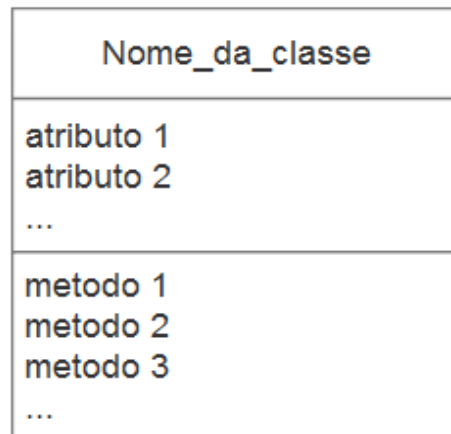
➤ Composição

➤ Classe associativa

➤ Generalização

➤ Interfaces

- ❖ Graficamente, as classes são representadas por rectângulos incluindo nome, atributos e métodos.



- ❖ Devem receber nomes de acordo com o vocabulário do domínio do problema.
- ❖ Devemos seguir 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(estado) dos objectos daquela classe.

➤ Visibilidade:

➤ **+** public: (público)

➤ **#** protected: (protegido)

➤ **-** private (privado)

➤ Exemplo

➤ **+** nome: String

## ➤ Métodos

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

### ➤ Visibilidade:

- **+** public: (público)
- **#** protected: (protegido)
- **-** private (privado)

### ➤ Exemplo

- **-** getNome(): String



- Diagrama de Classes
- Notação UML para classes

### ➤ Relacionamentos

- Associação
  - Agregação
- Composição
- Classe associativa
- Generalização
- Interfaces

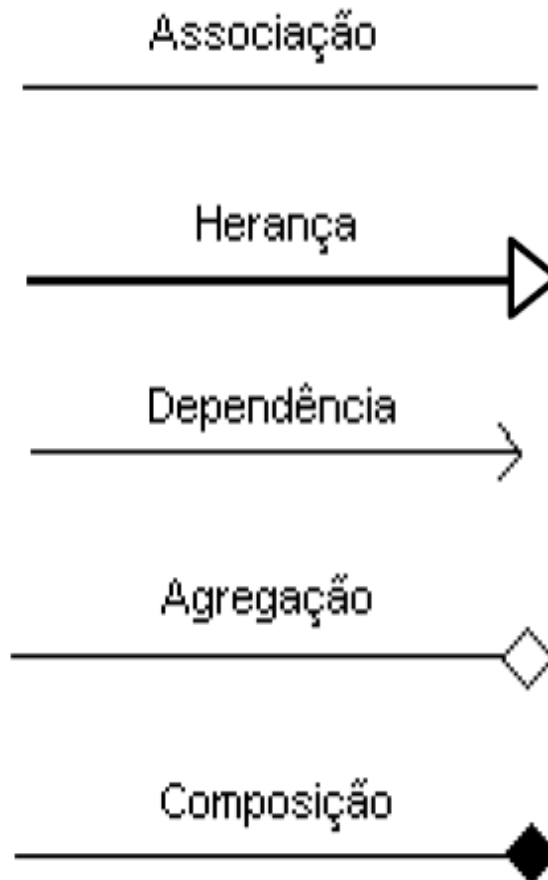
- ❖ Os relacionamentos possuem:
  - ❖ **Nome:** descrição dada ao relacionamento (faz, tem, possui, ministra,...)
  - ❖ **Sentido de leitura**
  - ❖ **Navegabilidade:** indicada por uma seta no fim do relacionamento
  - ❖ **Multiplicidade:** 0..1, 0..\*, 1, 1..\*,...
  - ❖ **Tipo:** associação (agregação, composição), generalização
  - ❖ **Papéis:** desempenhados por classes em um relacionamento
- ❖ **Navegabilidade e papéis** são opcionais para aumentar a clareza do diagrama.

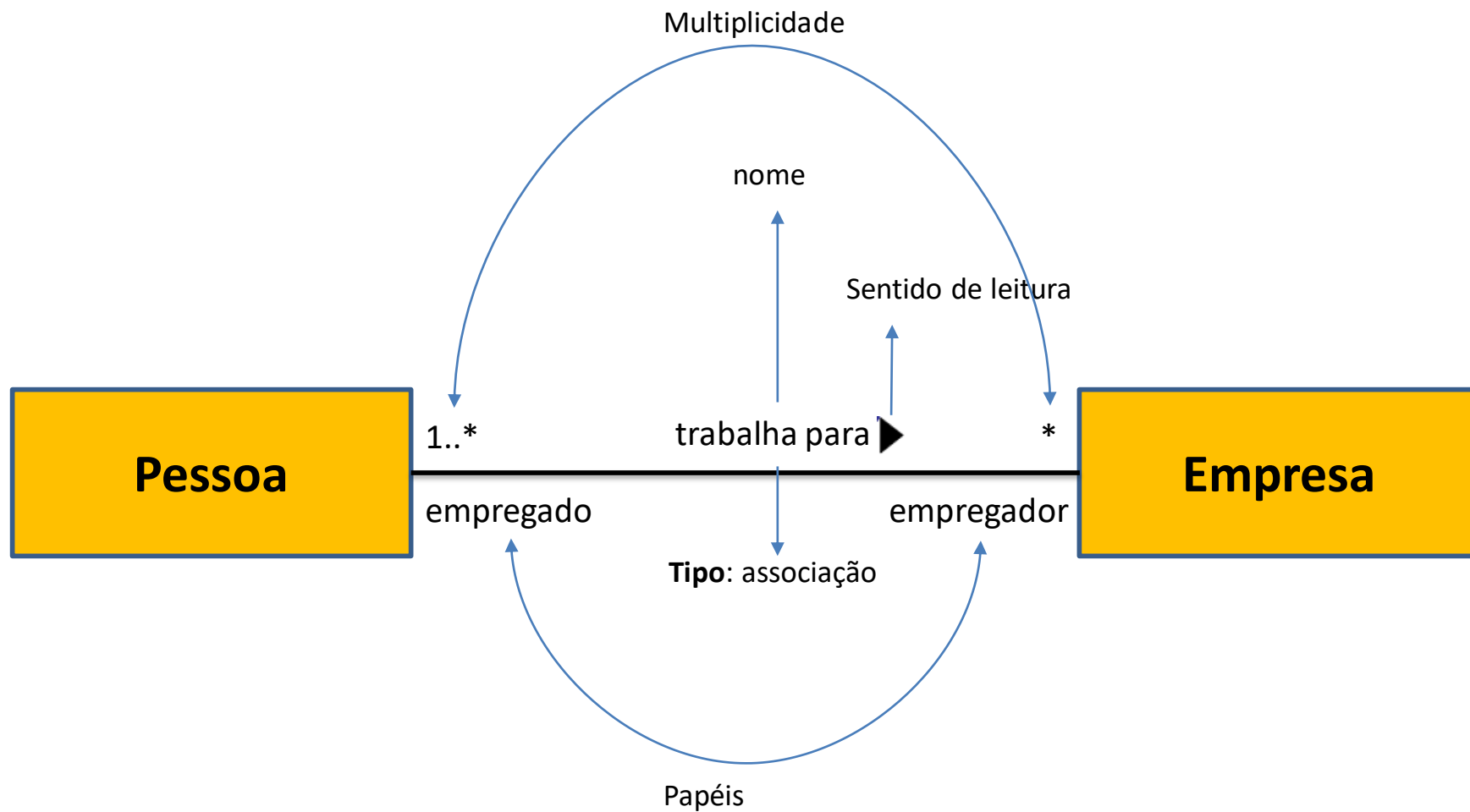
❖ Os tipos de relacionamentos são:

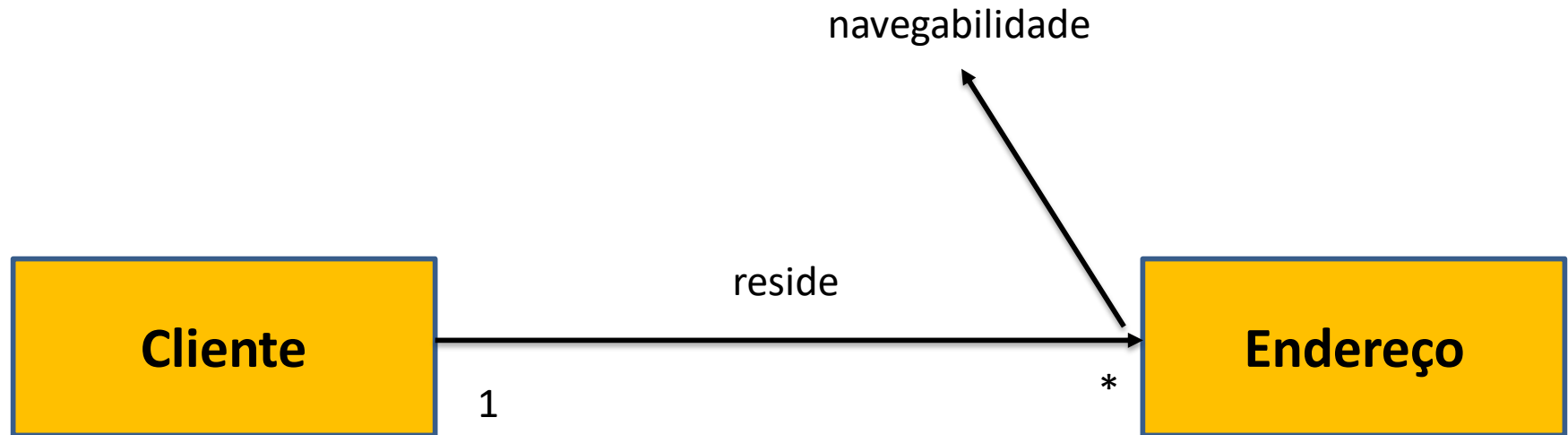
❖ **Associação**

- Agregação
- Composição

❖ **Especialização / Generalização**







- O cliente sabe quais são os seus endereços, mas o endereço não sabe a quais clientes pertence

- Diagrama de Classes
- Notação UML para classes
- Relacionamentos

- **Associação**

- Agregação
  - Composição
- Classe associativa
- Generalização
- Interfaces

- É um relacionamento estrutural que indica que os objectos de uma classe estão vinculados a objectos de outra classe
- Objectos podem **usar outros objectos**
  - **Comboio usa estrada de ferro**
- Uma associação é representada por uma linha sólida conectando duas classes.



```
public class Comboio {  
    private EstradaDeFerro estradaDeFerro;  
}
```

➤ Algumas das multiplicidades mais comuns

❖ **1**                      Exactly one

❖ **1..\***                  One or more

❖ **0..\***                  Zero or more

❖ **\***                      Many

❖ **0..1**                 Zero or one

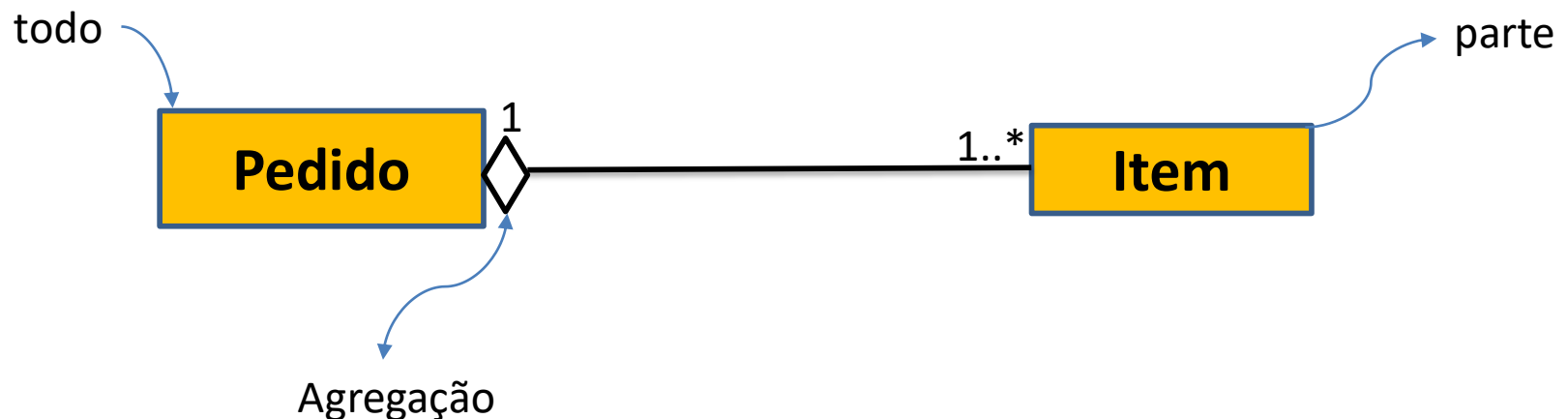
Multiplicidade





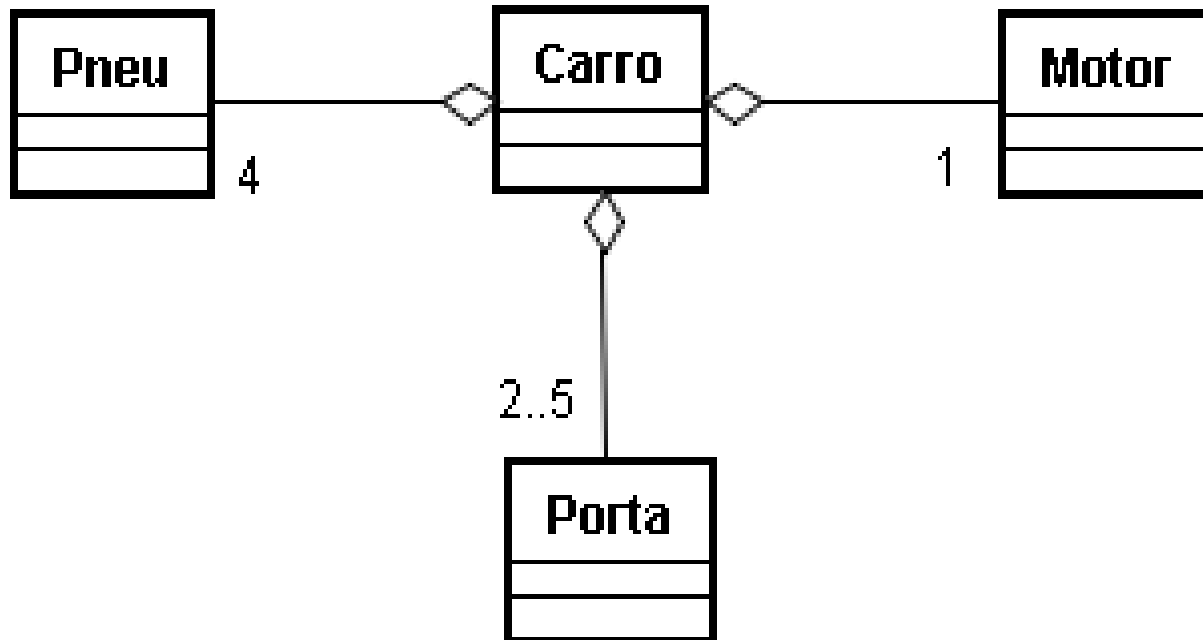
- Diagrama de Classes
- Notação UML para classes
- Relacionamentos
  - Associação
  - **Agregação**
  - Composição
  - Classe Associativa
  - Generalização
- Interfaces

- É um tipo especial de associação.
- Utilizada para indicar “todo-parte”.
- A classe filha faz sentido mesmo se a classe Pai deixar de existir.
- Se a classe Pai for apagada, a classe filha continuará a existir sem problemas.



➤ **Carro possui Pneu, Motor e Porta**

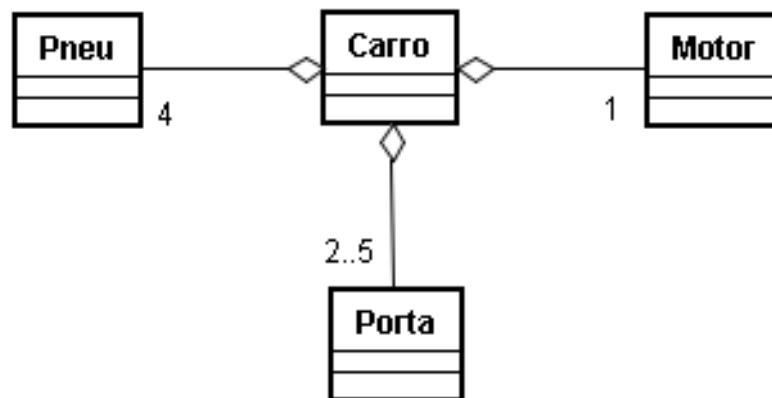
- ❑ Não são partes essenciais do Carro
  - Retirando as portas um carro continua a ser um carro.
  - Pneus/portas existem como objectos independentes



```
public class Carro {
    private Motor motor;
    private Porta portas[];
    private Pneu pneus[];
}
```

Pode ser implementado  
De mais de uma forma

```
public class Carro {
    private Motor motor;
    private Porta portas[];
    private Pneu pneu1;
    private Pneu pneu2;
    private Pneu pneu3;
    private Pneu pneu4;
}
```



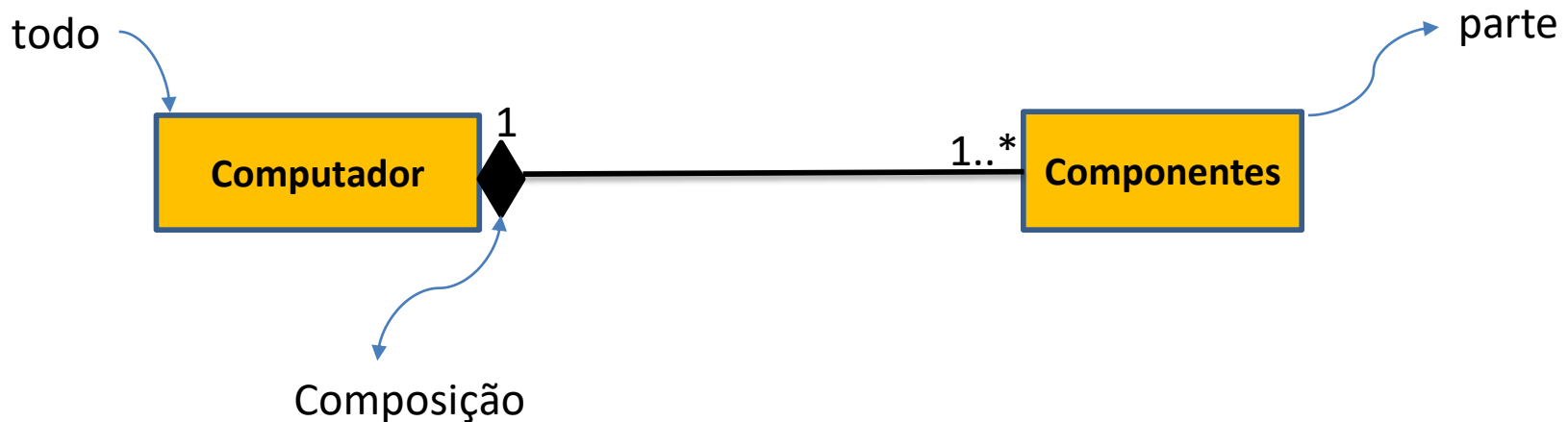
```
public class Motor {
}

public class Pneu {
}

public class Porta {
}
```

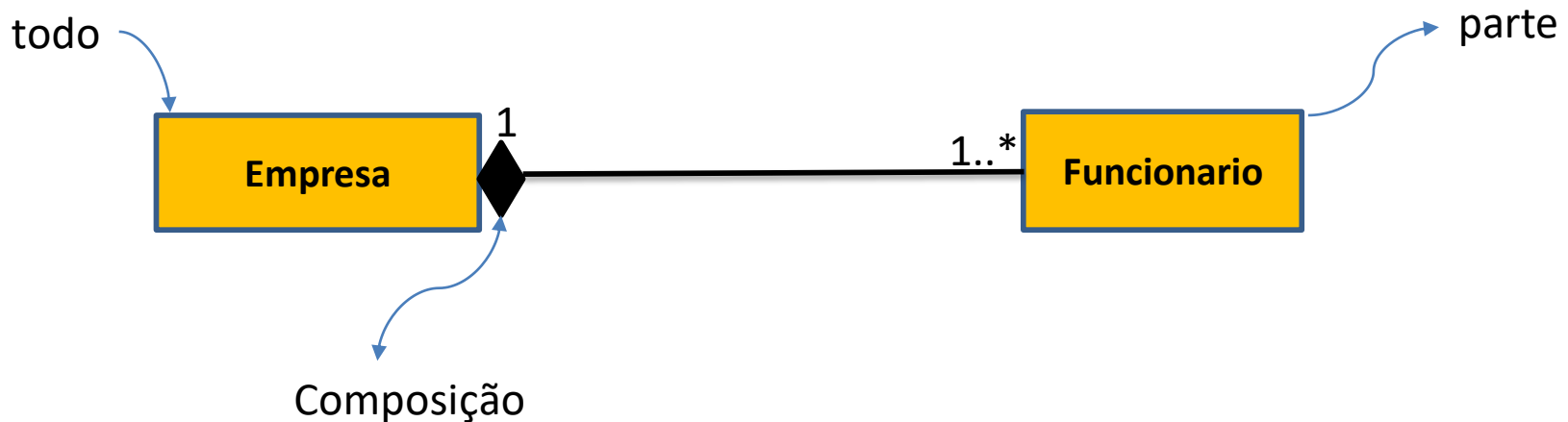
- Diagrama de Classes
- Notação UML para classes
- Relacionamentos
  - Associação
  - Agregação
  - Composição**
  - Classe Associativa
  - Generalização
- Interfaces

- É uma variante semanticamente mais “forte” da agregação.
- Os objetos “parte” só podem pertencer a um único objeto “todo” e têm o seu tempo de vida coincidente com o dele.
- Uma classe filha só faz sentido existir se uma classe pai existir.
- Se a classe pai é apagada automaticamente a classe filha é apagada também.

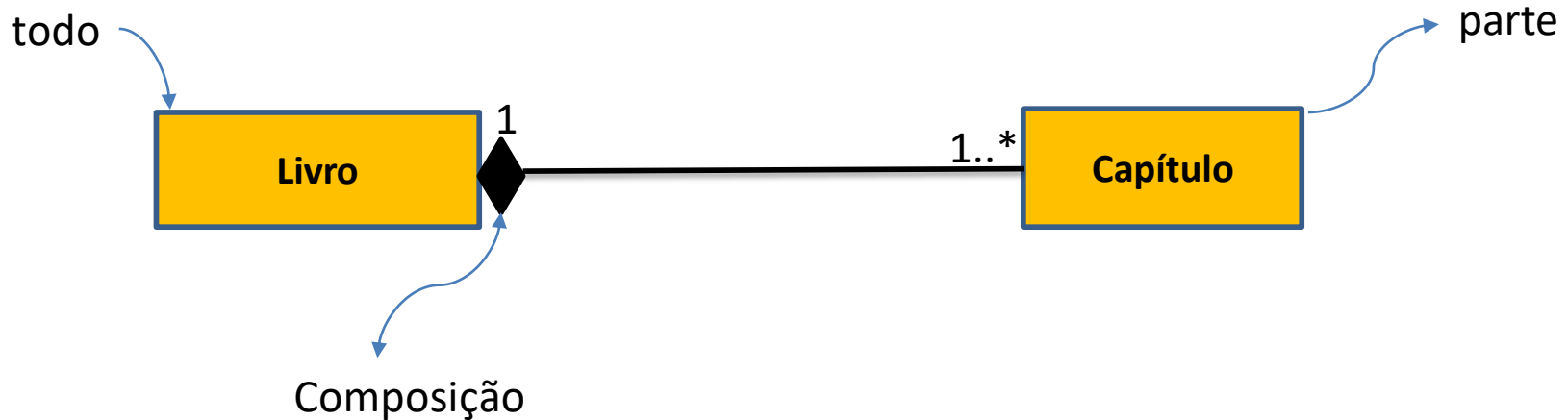


- ❖ Quando o “todo” morre todas as suas “partes” também morrem.

- ❖ Imaginando um sistema de recursos humanos , vamos supor as classes **funcionário** e **empresa**, onde funcionário é classe “parte” e empresa a classe “todo”.
- ❖ Nesse cenário não faz sentido ter funcionários, senão existir uma empresa onde eles possam trabalhar.
- ❖ Se a empresa deixar de existir automaticamente deixa de ter funcionários



- ❖ Um livro é composto por capítulos
  - ❑ Capítulo é parte essencial de livro
    - Se não existir capítulo, não existe livro
    - Capítulo não existe fora de livro
- ❖ **Livro é composto por 1 ou mais Capítulos**





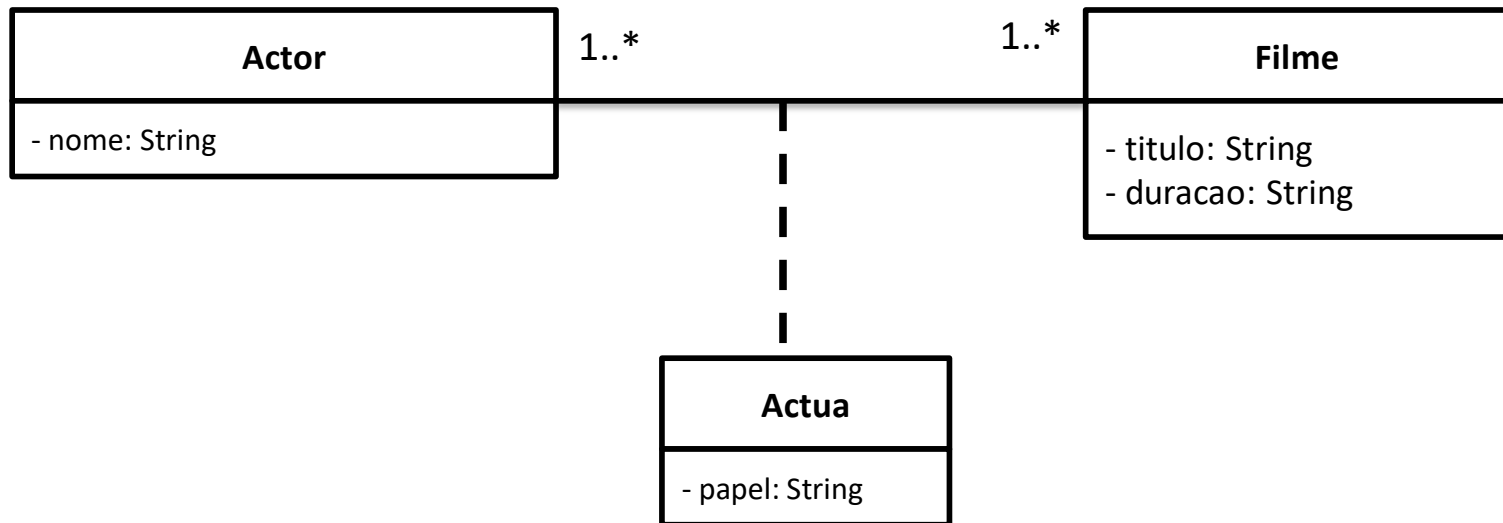


```
public class Capitulo {  
    private Livro livro;  
}
```

```
public class Livro {  
    private Capitulo[] capitulos;  
  
    public Livro(int qtdCapitulos){  
        capitulos = new Capitulo[qtdCapitulos];  
    }  
}
```

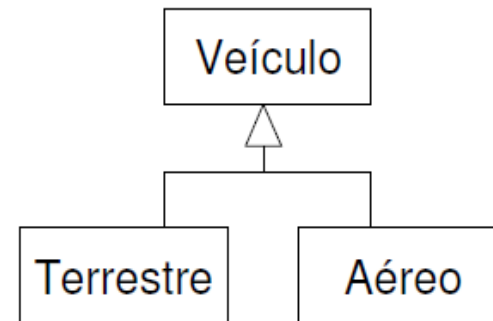
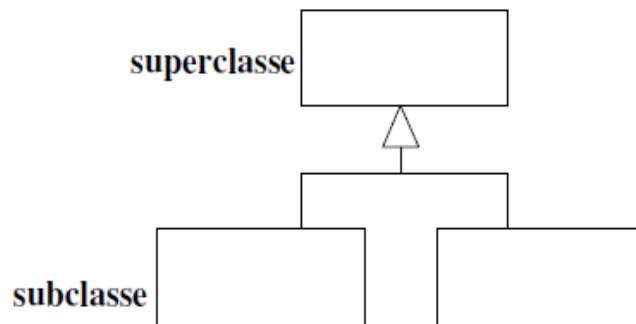
- Diagrama de Classes
- Notação UML para classes
- Relacionamentos
  - Associação
  - Agregação
  - Composição
- **Classe Associativa**
- Generalização
- Interfaces

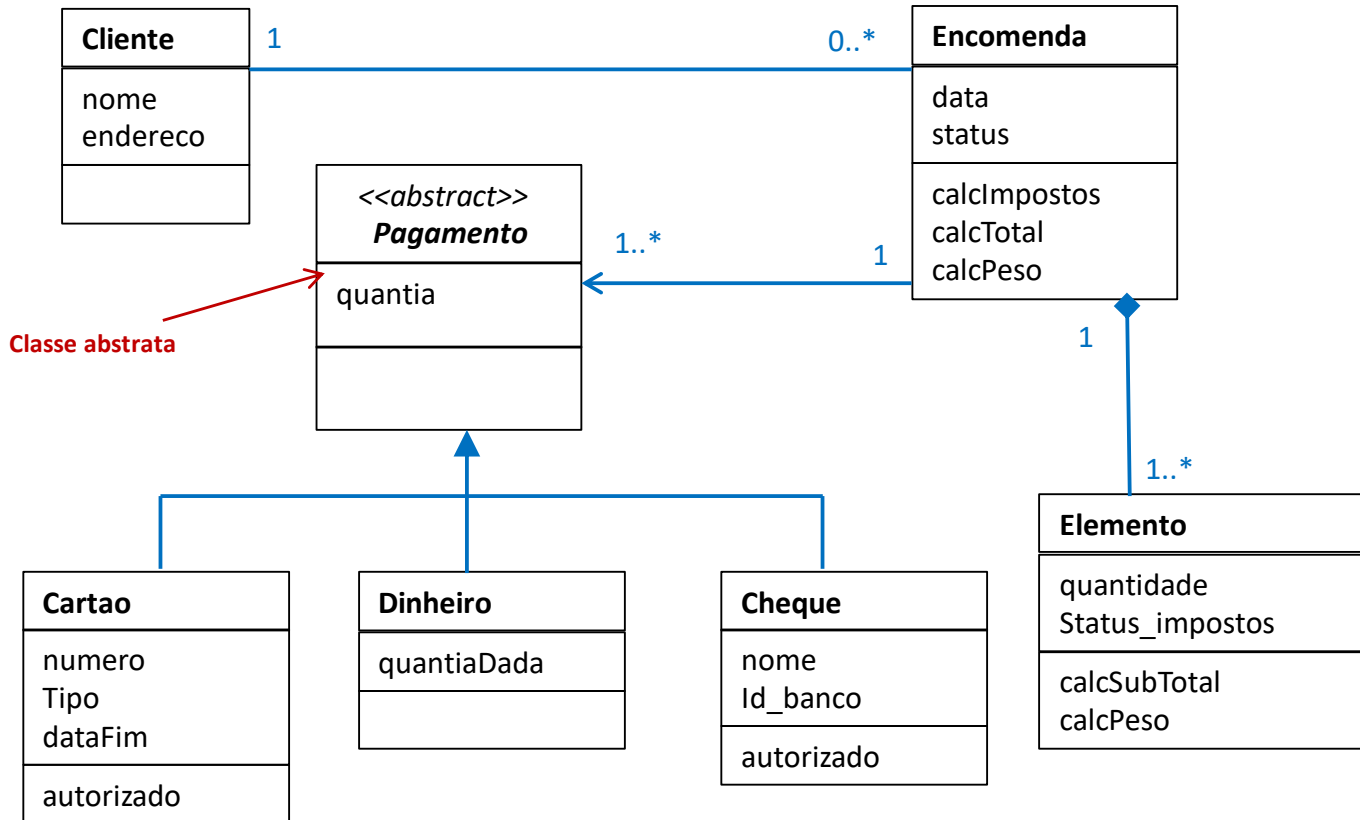
- ❖ Utilizada quando ocorrem associações que possuem multiplicidade muitos para muitos em todas as suas extremidades
- ❖ Armazena os atributos transmitidos pela associação
- ❖ Pode possuir os seu próprios atributos
- ❖ Representada por recta tracejada partindo do meio da associação até uma classe



- Diagrama de Classes
- Notação UML para classes
- Relacionamentos
  - Associação
    - Agregação
    - Composição
  - Classe Associativa
- **Generalização**
- Interfaces

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





- Diagrama de Classes
- Notação UML para classes
- Relacionamentos
  - Associação
    - Agregação
    - Composição
  - Generalização
- Interfaces

- ❖ Tipo especial de classe a qual não pode ser instanciada.
- ❖ Serve apenas para especificar operações externamente visíveis para uma outra classe implementar

