



**PUC Minas**

CIÊNCIA DA COMPUTAÇÃO

Engenharia de Software I

Profa: Luciana Mara F. Diniz

# Diagrama de Classes (UML)

<i><b>Empregado</b></i>
- nome : String - sobrenome : String - cpf : String

# CASOS DE USO ↔ CLASSES

- Interdependência entre os dois modelos

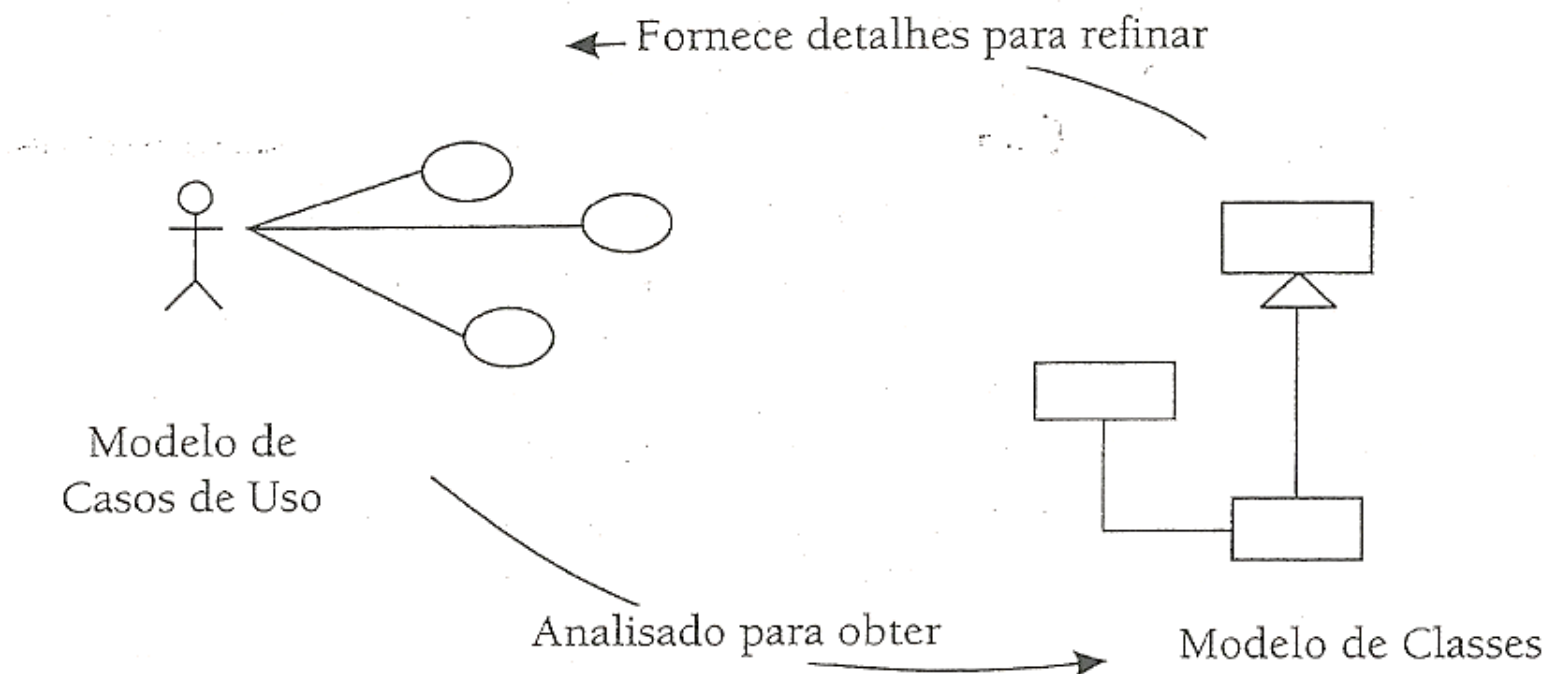





Figura 5-46: Interdependência entre o modelo de casos de uso e o modelo de classes.

# ANÁLISE REQUISITOS


CÓDIGO	NOME REQUISITO	DESCRIÇÃO
RF 04	Cadastrar veículos	Atributos: placa, fabricante, proprietário, espécie, tipo, data de aquisição, cidade da placa, RENAVAM, número do motor, ano de fabricação, ano do modelo.



 Veículos / Novo / VW CROSSFOX

Placa:  

Fabricante:

Proprietário:  

Espécie:

Tipo:

Data de aquisição:


Cidade da placa:

RENAVAM:

Número do motor:

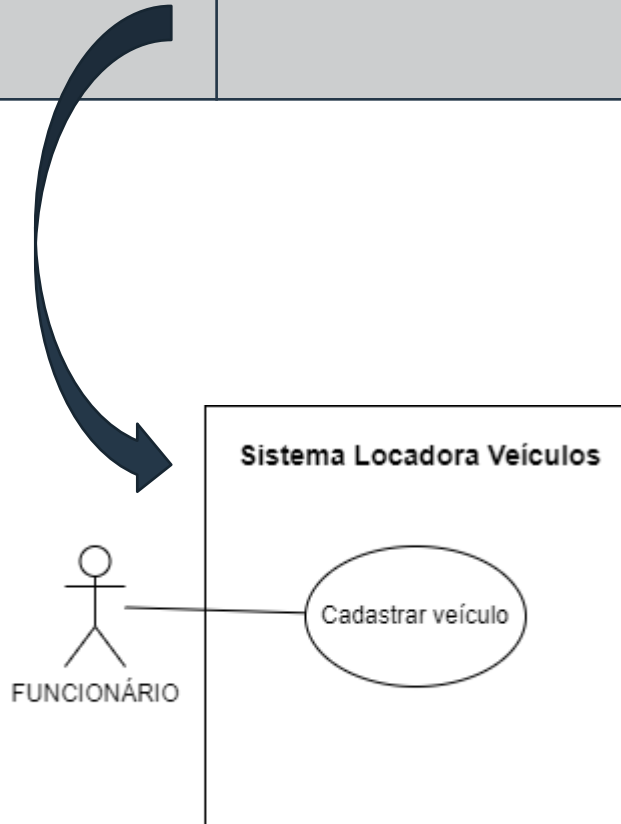
Ano de fabricação:

Ano do modelo:

 Salvar

# DIAGRAMA CASOS DE USO

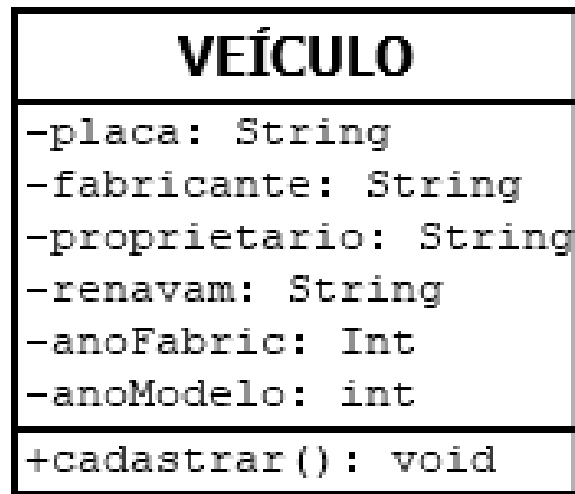
CÓDIGO	NOME REQUISITO	DESCRIÇÃO
RF 04	Cadastrar veículos	Atributos: placa, fabricante, proprietário, espécie, tipo, data de aquisição, cidade da placa, RENAVAM, número do motor, ano de fabricação, ano do modelo.



ID e NOME	CSU 04 – CADASTRAR VEÍCULO
RESUMO	Os dados para cadastro de um veículo são placa, modelo, ano, data de aquisição e status (alugado, disponível, conserto).
PRÉ-CONDIÇÕES	<b>Proprietário cadastrado no sistema (CSU 02)</b>
PRIORIDADE	Essencial
ATOR PRINCIPAL	Funcionário
ATOR SECUNDÁRIO	Não se aplica.
FLUXO PRINCIPAL	1. O funcionário abre a tela com o formulário de cadastro. 2. O funcionário preenche os campos do formulário. 3. O funcionário clica no botão de Cadastrar.
FLUXO ALTERNATIVO	2. Funcionário sai da tela sem salvar os dados. a) O sistema emitirá uma mensagem se deseja realmente sair sem salvar incluindo botões com as opções “sim” ou “não”. b) O funcionário escolhe uma opção. c) Se sim, continua no passo 2 do fluxo principal.  2. Funcionário cancela o cadastro. a) O funcionário clica no botão Cancelar para sair do cadastro.
FLUXO DE EXCEÇÃO	3. Campos obrigatórios ficam em branco a) O sistema mostrará na tela os itens obrigatórios do formulário que devem ser preenchidos, assinalados de vermelho. b) Volta ao passo 2 do fluxo principal.
PÓS-CONDIÇÕES	Veículo cadastrado no sistema.
REGRAS DE NEGÓCIO	Não se aplica.

# DIAGRAMA DE CLASSES

CÓDIGO	NOME REQUISITO	DESCRIÇÃO
RF 04	Cadastrar veículos	Atributos: placa, fabricante, proprietário, espécie, tipo, data de aquisição, cidade da placa, RENAVAM, número do motor, ano de fabricação, ano do modelo.



# DIAGRAMA DE CLASSES

- O modelo de classes de especificação apresenta as classes identificadas e detalhamento do modelo de classes (**COMO** o sistema deve fazer);
- O modelo de classes de **implementação**: posterior implementação das classes em alguma linguagem OO (Java, PHP, Python, C++, C#).
- Um modelo de Classes possui (assim como DCUs):
  - Diagrama de classes, e
  - Descrição textual;

# DIAGRAMA DE CLASSES

- O **diagrama de classes** é utilizado na construção do modelo de projetos, além da redefinição do modelo de análise de requisitos e usado na implementação.
  - Mostra um conjunto de classes e suas relações.

## ELEMENTOS DE UM DIAGRAMA DE CLASSES:

- **Classes**
- **Relacionamentos**
  - Associação
    - Multiplicidades
    - Agregação e Composição
  - Generalização (herança)

# DIAGRAMA DE CLASSES

- **CLASSE**

- Uma classe é representada por uma “caixa” com três compartimentos (níveis) exibidos. Em outras palavras, as classes são representadas por retângulos incluindo **nome, atributos e métodos**.

- Os níveis são:

- 1º nível – nome da classe
- 2º nível – atributos
- 3º nível – funções/operações/métodos





# DIAGRAMA DE CLASSES

- **ATRIBUTOS**

SÍMBOLO	NOME	SIGNIFICADO
+	público	visível em qualquer classe
#	protegido	qualquer descendente pode usar
-	privado	visível somente dentro da classe

- Ex.:



- **SINTAXE:** visibilidade + NomeAtributo : tipo

# DIAGRAMA DE CLASSES

- OPERAÇÕES - métodos / funções

SÍMBOLO	NOME	SIGNIFICADO
+	público	visível em qualquer classe
#	protegido	qualquer descendente pode usar
-	privado	visível somente dentro da classe

- Ex.:

<b><i>Empregado</i></b>
+ <i>vencimento()</i> : double

- **SINTAXE:** visibilidade NomeMétodo() : tipoRetorno

\* Os parâmetros podem aparecer ou serem ocultados

# ASSOCIAÇÕES

- **Associações** – representam **relacionamentos entre classes**;
  - Permitem que as classes compartilhem informações entre si;
  - Descreve um vínculo que ocorre normalmente entre os objetos de uma ou mais classes.
- OBS.: **Objetos são instâncias das classes.**



# ASSOCIAÇÕES

- **Multiplicidades:**

- permitem representar a informação dos limites **inferior** e **superior** da quantidade de objetos aos quais outro objeto pode estar associado.

→ **OBSERVAÇÃO:** é obrigatório colocar no diagrama!!



# ASSOCIAÇÕES

CARDINALIDADE  
=  
BANCO DE DADOS

- Símbolos de multiplicidade:

Nome	Simbologia
Apenas um	1
Muitos	*
Um e somente um	1..1
Zero ou muitos	0..*
Um ou muitos	1..*
Zero ou um	0..1
Intervalo específico	$L_i..L_s$ 2..10

Li – limite inferior  
Ls – limite superior

# ASSOCIAÇÕES

- Multiplicidades

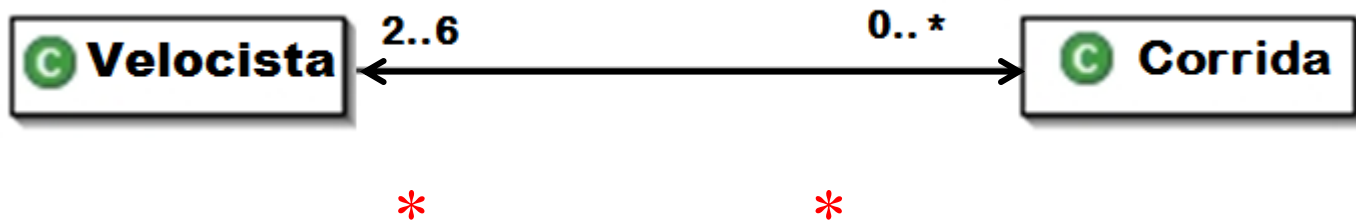
- Exemplo 1:

- um objeto da classe **Cliente** que está associado a vários objetos da classe **Pedido** e um objeto da classe **Pedido** está associado a somente um **Cliente**.



# ASSOCIAÇÕES

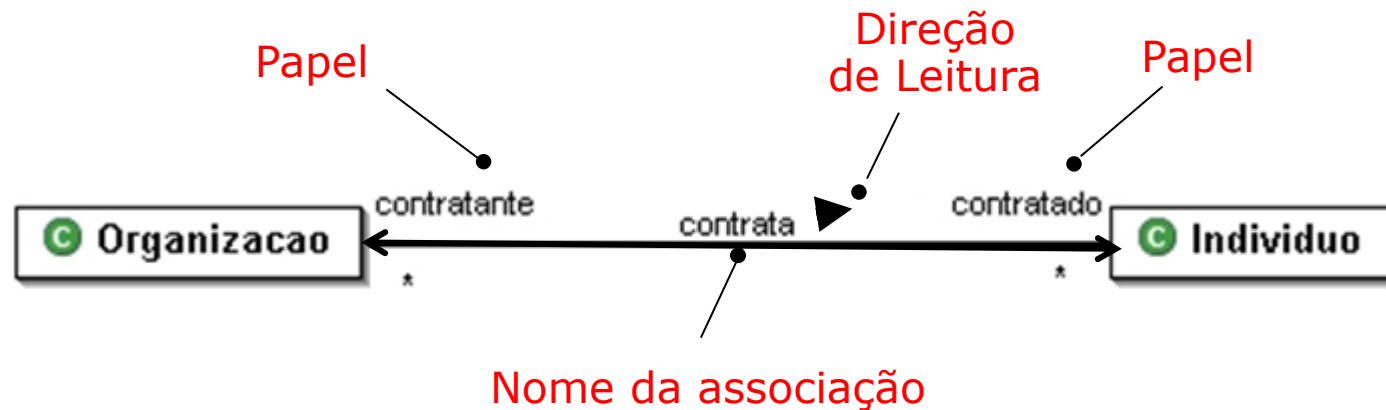
- Multiplicidades
- Exemplo 2:
  - informações sobre **Velocistas e Corridas** nas quais eles participam, com intervalo específico (limites inferiores e superiores).



# ASSOCIAÇÕES

- Três recursos de notação (**OPCIONAIS**):

- nome de associação,
- direção de leitura e
- papel.

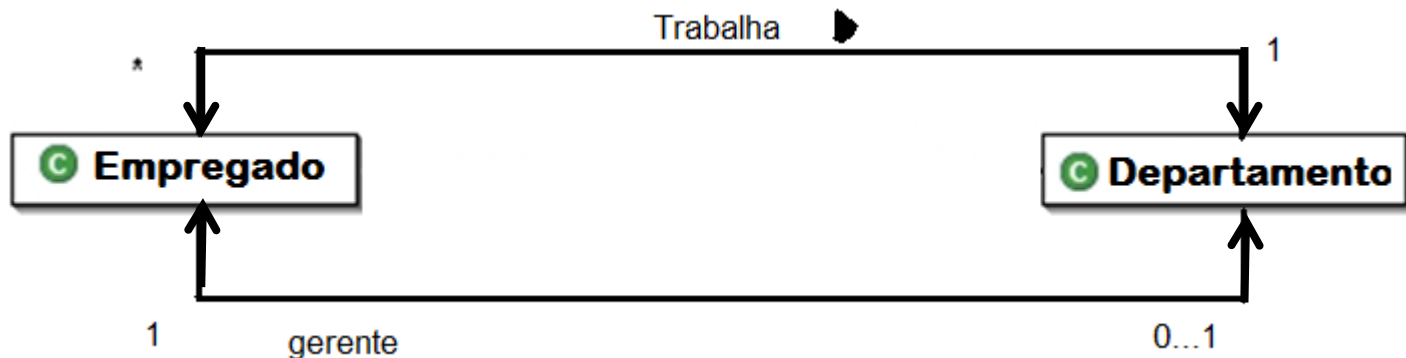


(Deve-se verificar a necessidade do uso destes recursos para **não “poluir”** o diagrama).



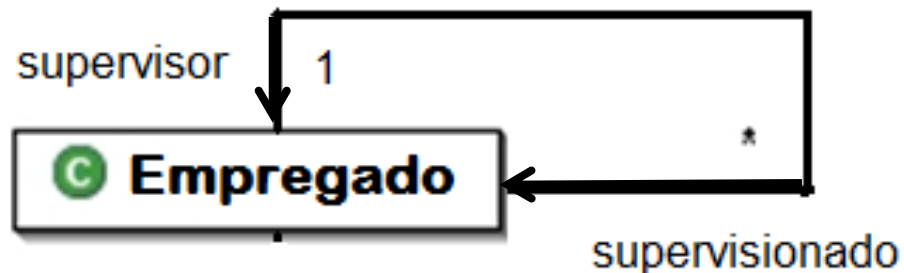
# ASSOCIAÇÕES

- Pode haver duas associações (ou mais) definidas entre duas classes no diagrama de classes.
- Neste caso há duas **associações diferentes**, com **multiplicidades diferentes**, embora as classes envolvidas sejam as mesmas.



# ASSOCIAÇÕES

- Uma associação reflexiva (ou autoassociação) associa objetos da mesma classe.
- Cada objeto tem um **papel distinto** nessa associação.



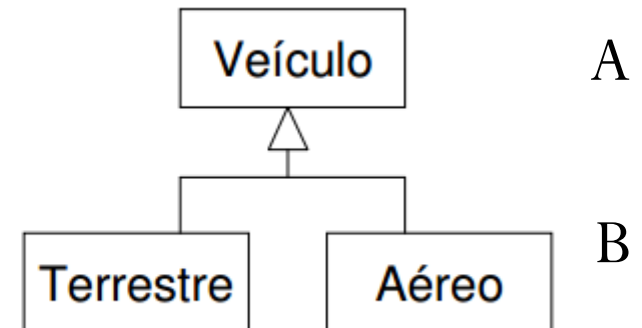
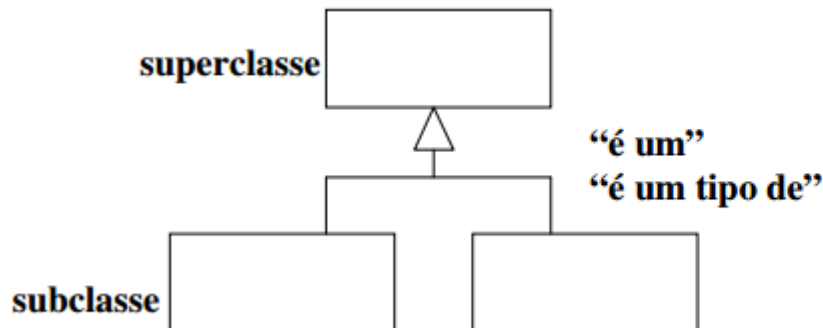
# GENERALIZAÇÃO

- **Relacionamento entre classes.**
- Estes relacionamentos podem ser de **generalidade ou herança**, como também é conhecido.
- Entre duas classes A e B, se A é uma generalização de B, então B é uma especialização de A.

generalizo



especializo



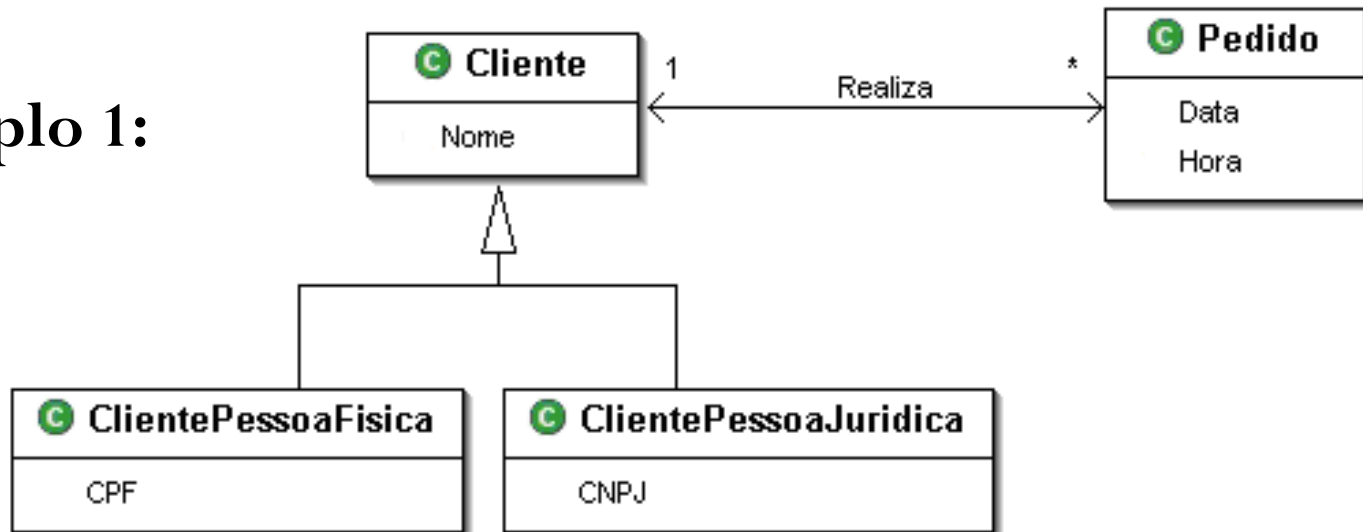
# GENERALIZAÇÃO

- Os termos para denotar o relacionamento de herança são variados:
  - **Superclasse e subclasse;**
  - Classe base e classe herdeira;
  - Supertipo e subtipo.
  - Classe mãe e classe filha.
- A herança é representada na UML por **uma flecha partindo da(s) subclasse(s) em direção à superclasse.**

# GENERALIZAÇÃO

- Os atributos, operações e também associações são herdados pelas subclasses (**ver exemplo da associação REALIZA abaixo**).
- Por outro lado, se existe uma associação que não é comum a todas as subclasses, mas somente uma ou outra, então a associação deve ser representada somente entre a classe em questão.

- **Exemplo 1:**

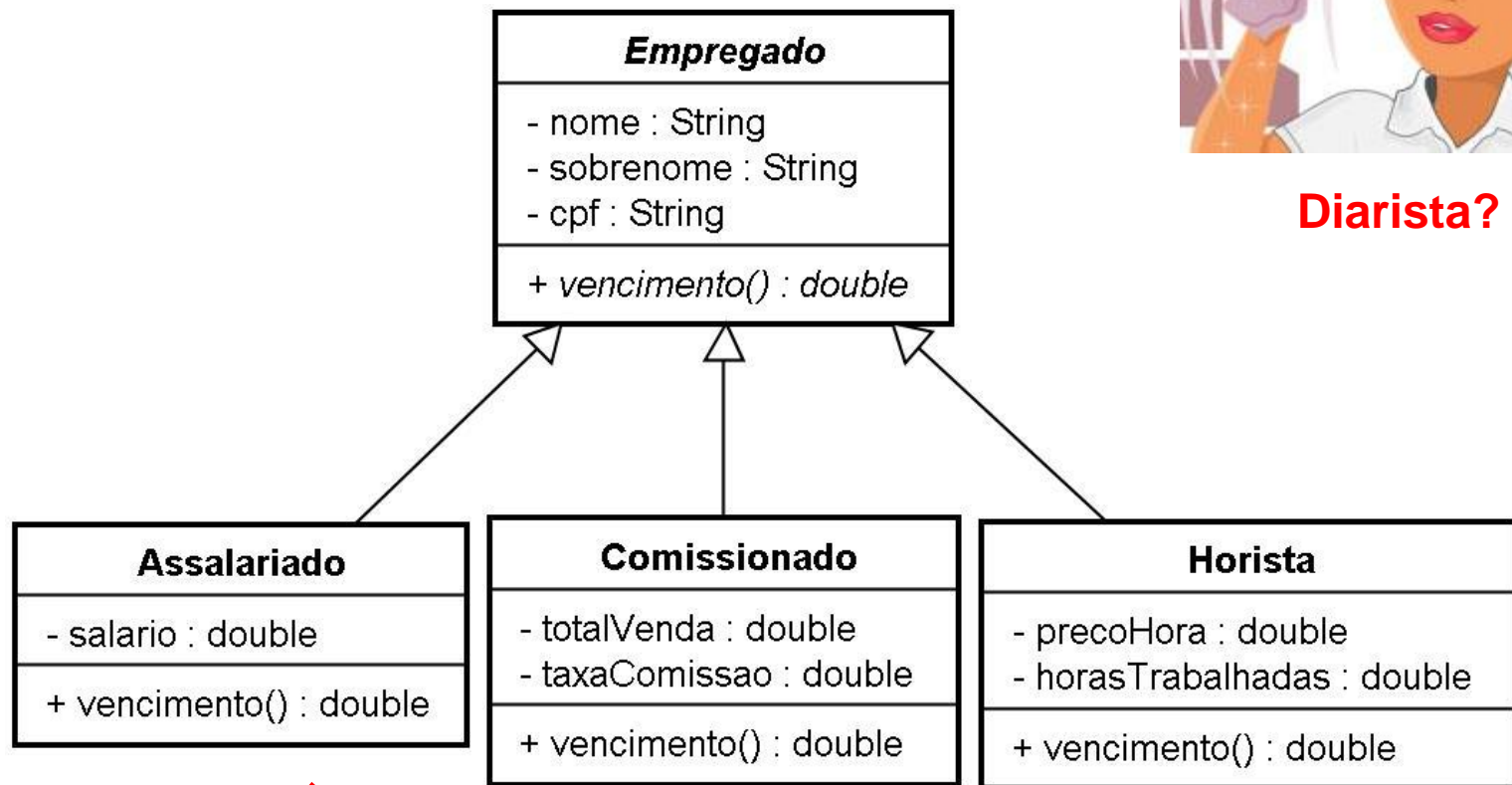


# GENERALIZAÇÃO

## Exemplo 2:



**Diarista?**



P O L I M O R F I S M O

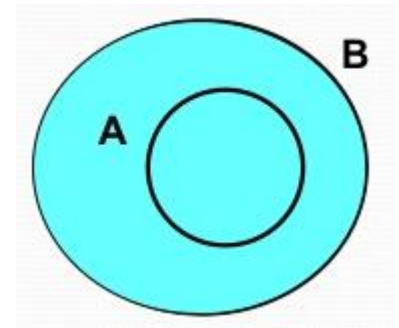
# AGREGAÇÃO e COMPOSIÇÃO

- É um tipo especial de ASSOCIAÇÃO entre duas ou mais classes.
- Demonstra que **as informações de uma classe (TODO) precisam ser COMPLEMENTADAS pelas informações contidas em uma outra classe (PARTE).**
- Em outras palavras, indica que:
  - um **objeto está contido no outro** ou um **objeto contém o outro.**
- A UML define 2 tipos de relacionamentos **TODO-PARTE**:
  - Agregações, e
  - Composições.

# AGREGAÇÃO e COMPOSIÇÃO

## - Características:

- se um objeto A é parte de um objeto B, o objeto B não pode ser parte do objeto A, pois o objeto B é o TODO.



## - Identificando estas associações

- Pergunta: “B tem um ou mais A”? Ou “A é parte de B”?
- Se sim a estas perguntas, provavelmente há um relacionamento todo-parte, onde B é o **todo** e A é a **parte**.
  - Ex.: “**B** tem um ou mais **A**”? Ou “**A** é parte de **B**”?  
**CARRO** tem uma (ou mais) **RODA**? Ou **RODA** é parte de **CARRO**?



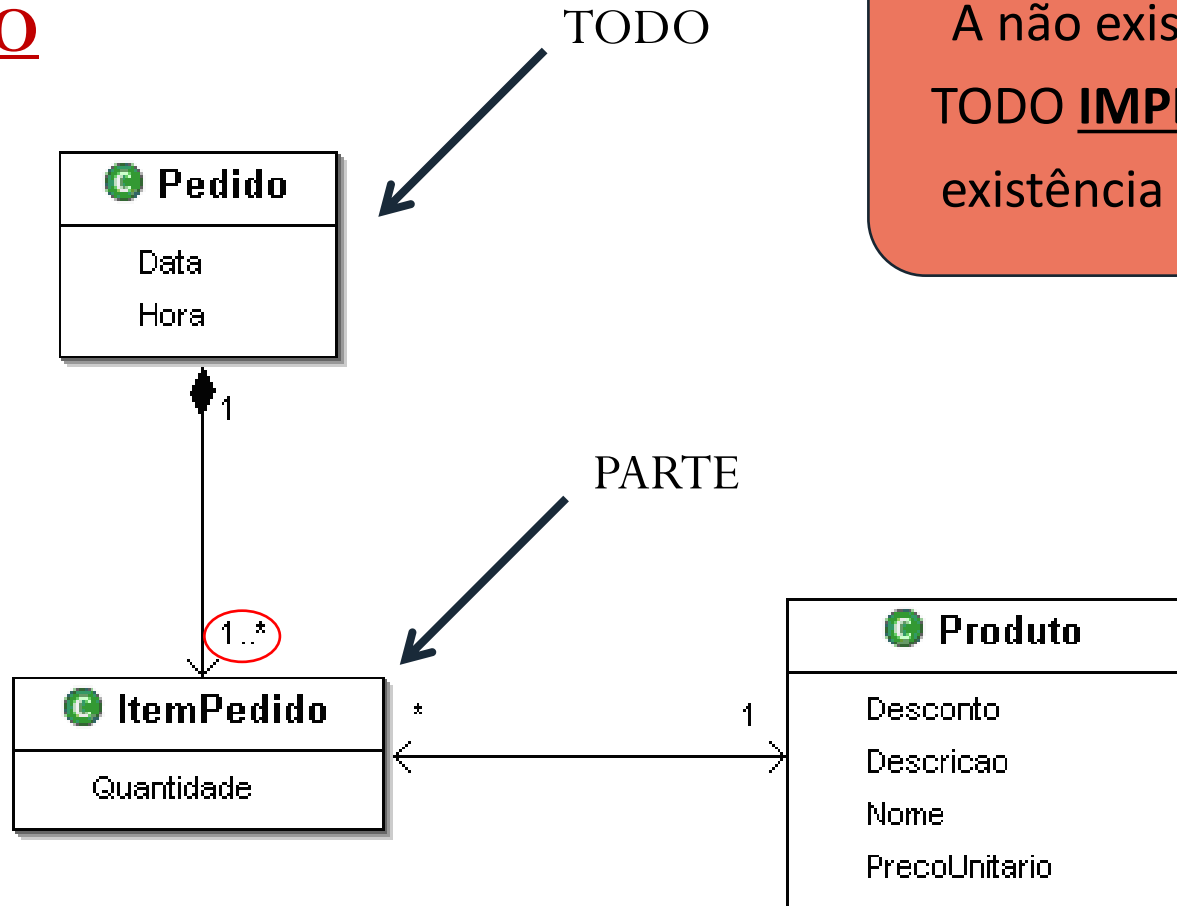
# COMPOSIÇÃO

- Uma composição é representada na UML com uma **linha que conecta as classes relacionadas com um losango preto**, perto da classe que representa o TODO).
- Identifica-se primariamente a classe TODO para representar a simbologia da forma correta.
- SÍMBOLO:



# COMPOSIÇÃO

- EXEMPLO**



A não existência do **TODO** **IMPLICA** a não existência da **PARTE**.

- Na **composição**, os objetos parte pertencem a um único **TODO**.

# COMPOSIÇÃO

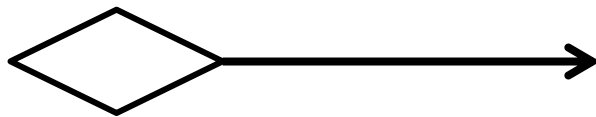
”

*/\* Num relacionamento de Composição podemos entender que classes que são compostas por outras precisam destas para “viver”, para “existir”.*

- Na **composição** o objeto todo é responsável por criar (**instanciar objetos**) suas partes.
- Além disso, as informações da classe PARTE são usadas para **complementar o entendimento** da classe TODO.

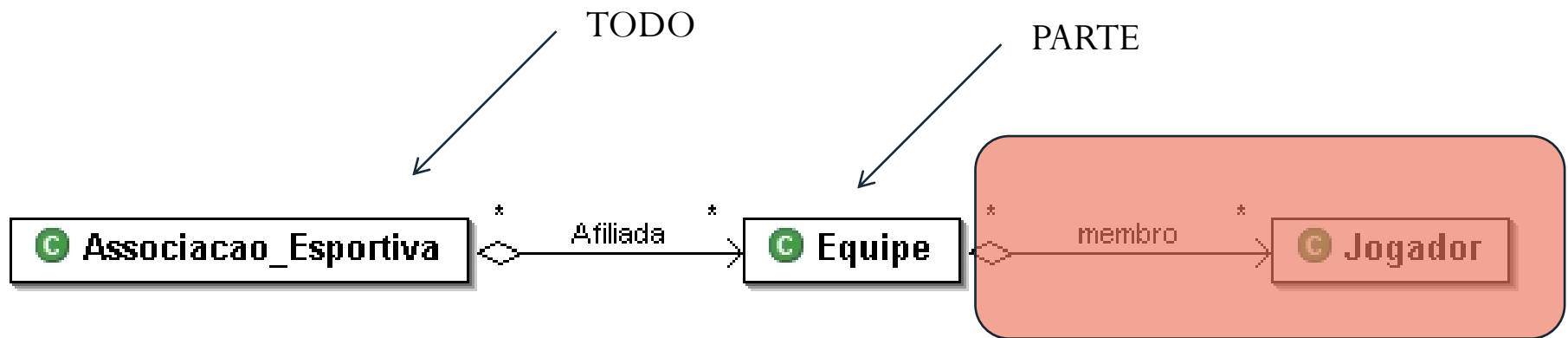
# AGREGAÇÃO

- Uma agregação é representada na UML por **uma linha que conecta as classes relacionadas com um losango sem preenchimento**, perto da classe que representa o todo.
- Identifica-se primariamente a classe TODO para representar a simbologia da forma correta.
- SÍMBOLO:



# AGREGAÇÃO

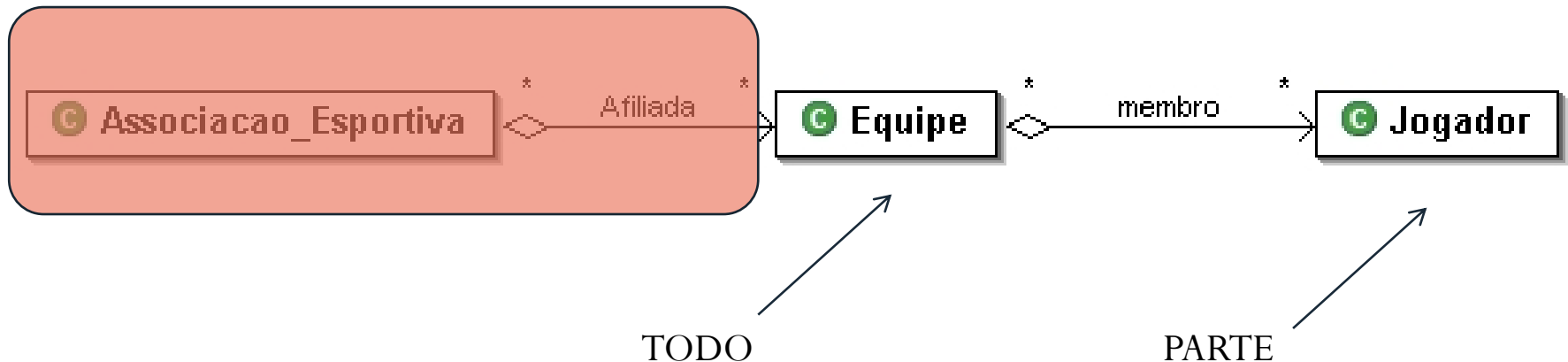
- EXEMPLO



- Na **agregação**, a não existência de um objeto TODO não implica necessariamente a não existência do objeto parte.

# AGREGAÇÃO

- EXEMPLO



- Na **agregação**, a não existência de um objeto TODO não implica necessariamente a não existência do objeto parte.

# AGREGAÇÃO



*/\* Num relacionamento de Agregação, podemos entender que classes que são agregadas por outras não precisam destas para “viver”, para “existir”.*

- Na agregação diferentemente da composição, **o objeto todo recebe as instâncias do objeto parte já prontas**, ou seja, ele não é o responsável por sua criação (instanciação).
- Desta forma, as **instâncias dos objetos parte são criadas fora da classe** todo e agregadas por meio de um método.

# AGREGAÇÃO e COMPOSIÇÃO

- **HIERARQUIA**

- Tanto a **agregação** como a **composição** podem se estender por vários **níveis**.
- Neste caso, tem-se uma **HIERARQUIA DE AGREGAÇÕES** ou uma **HIERARQUIA DE COMPOSIÇÕES**.

- **Exemplo:**

- Pode haver um objeto A que seja composto de B, e que este último seja composto por objetos C e assim por diante...



# AGREGAÇÃO e COMPOSIÇÃO


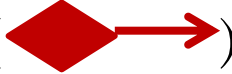

## → CONSIDERAÇÕES

- Sempre que uma classe TODO for consultada, além de suas informações, devem ser apresentadas também as informações da classe PARTE.

Atenção

- A agregação e a composição, podem, **em muitos casos**, serem substituídas por uma **associação binária simples**, dependendo de **QUEM FAZ A MODELAGEM** e do **CONTEXTO DO SOFTWARE**.

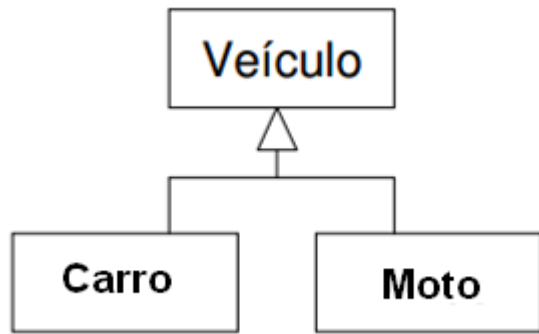
# ASSOCIAÇÃO X TODO-PARTE

- **COMO SABER QUAL USAR NA MODELAGEM? DICAS...**
- **1ª pergunta:** as informações de uma classe são necessárias para complementar as informações de outra classe?
  - **NÃO** → ASSOCIAÇÃO (  )
  - **SIM** → identificar as classes **TODO** e **PARTE**
    - **2ª pergunta:** a classe PARTE precisa da classe TODO para existir?
    - **3ª pergunta:** se não existir a classe TODO a classe parte não existe também?
      - Sim para as duas → COMPOSIÇÃO (  )
      - Não para as duas → AGREGAÇÃO (  )

# HERANÇA OU TODO/PARTE ?

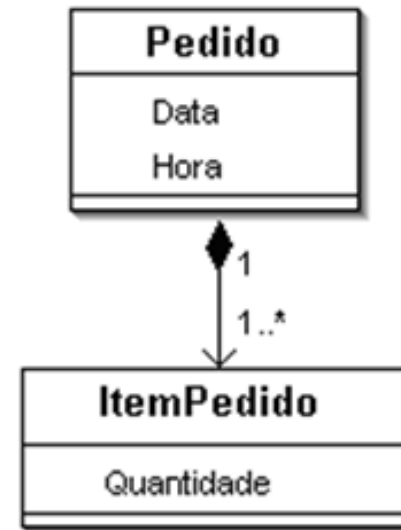
## GENERALIZAÇÃO (HERANÇA)

- É UM(A);
- É UM(A) TIPO DE;



## COMPOSIÇÃO/AGREGAÇÃO

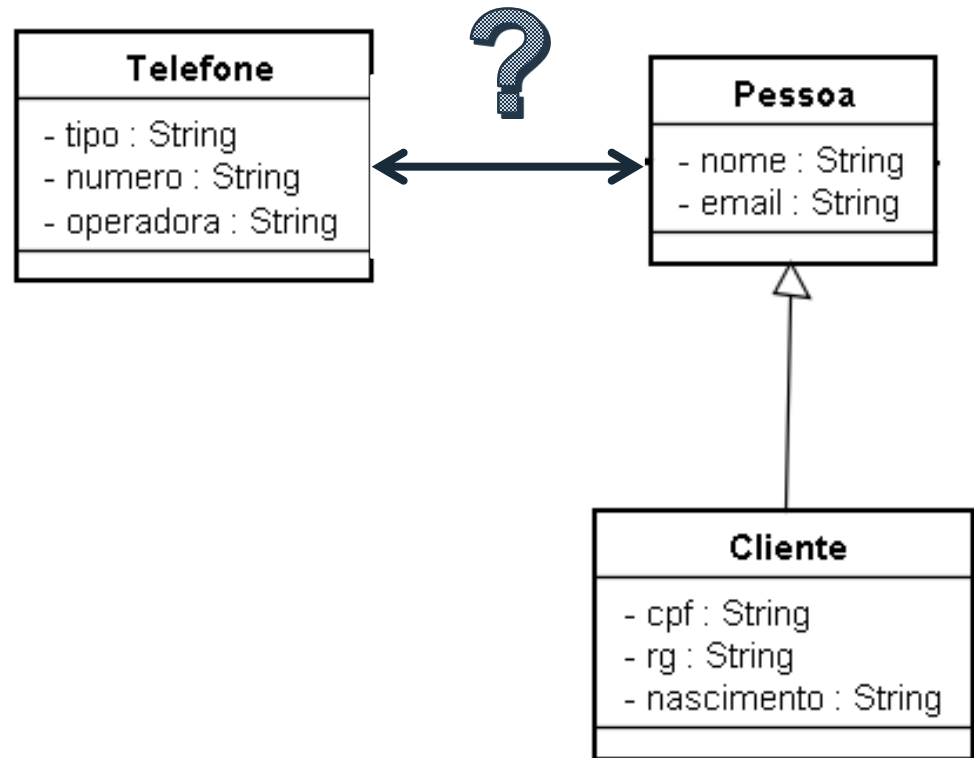
- TEM UM(A);
- É PARTE DE UM;



# Associação ou Composição/Agregação?

→ Associação

- CONTEXTO
- Sistema de Gestão para Transportadora

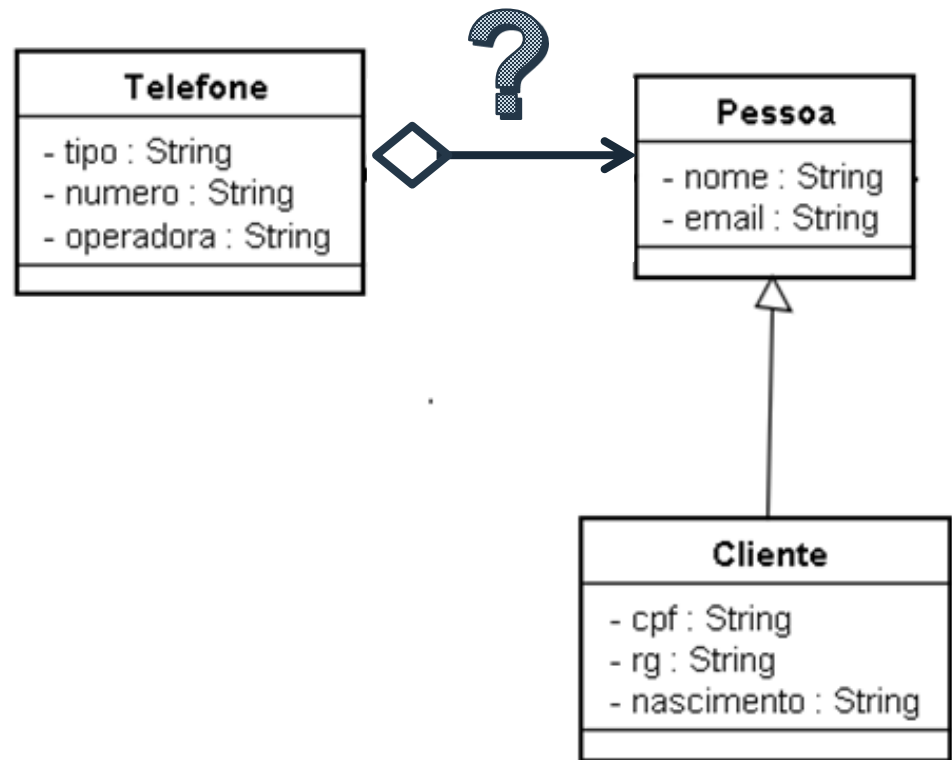


# Associação ou Composição/Agregação?

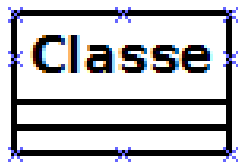
- A função principal de uma associação do tipo TODO/PARTE é **identificar a obrigatoriedade de uma complementação das informações** de uma classe TODO por uma classe PARTE quando for consultada.

Exemplo 2:

- Sistema de Gestão para Telefonia Móvel**



# SÍMBOLOS – DIAGRAMA DE CLASSE



Classe



Agregação

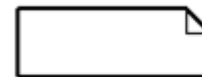


Composição



Associação  
(com multiplicidade e  
direção de leitura)

Generalização  
(herança)



Comentário

# Técnica para IDENTIFICAR classes

## CONCEITOS/TERMOS

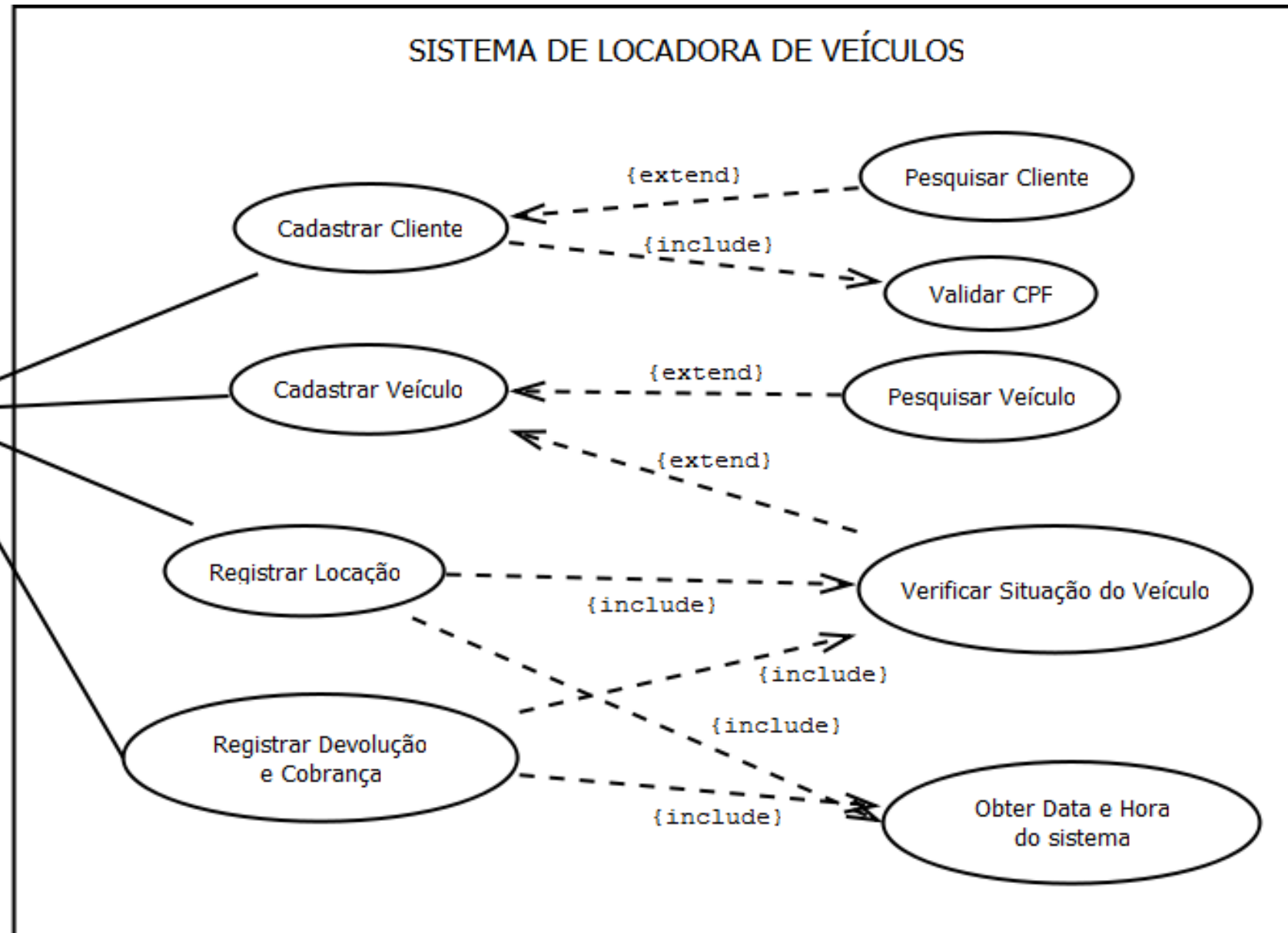
- **Conceitos concretos**, como edifícios, carros, salas de aula, etc.
- **Papéis desempenhados por seres humanos**, como professores, alunos, empregados, clientes, etc.
- **Eventos**, ou seja, ocorrências em uma data e hora particulares como reuniões, aulas, etc.
- **Lugares (áreas)**, como salas de aula, escritórios, etc.
- **Organizações**, ou seja, coleções de pessoas ou de recursos, como departamentos, projetos, campanhas, turmas, etc.
- **Conceitos abstratos**, isto é, princípios ou ideias não tangíveis, como vendas, reservas, inscrições, etc.

# PRATICANDO... EXERCÍCIO

- Fazer **Diagrama de CLASSES** para um de **Sistema Locação Veículos** (já fizemos diagrama de Casos de Uso deste cenário – próximo slide):
- Em uma locadora de veículos o funcionário da locadora cadastra e mantém os dados dos **veículos** (**placa, modelo, ano, status, chassi, cor, fabricante, data aquisição**) além de cadastrar e manter os dados de **clientes** (**nome, CPF, número da carteira de habilitação, endereço, telefone, data de nascimento**). O status de um carro pode assumir os valores disponível ou alugado. Também deve ser registrada a locação dos veículos, cadastrando dados do contrato de **locação** (**número do contrato, data locação, veículo, quilometragem, cliente, valor da diária, dados do seguro**). Além disso, devem ser cadastrados dados referente à **devolução e cobrança, registrando informações** (**número do contrato, data de devolução, veículo, quilometragem, total de diárias e valor total do pagamento**). As datas utilizadas para cadastro de Locação e Devolução devem ser obtidas do sistema operacional. Deve ser possível buscar o veículo locado tanto pelo nome do cliente tanto pela placa do carro. O valor total deve ser calculado a partir do total de diárias, valor do seguro/dia, quilometragem e situação do veículo.

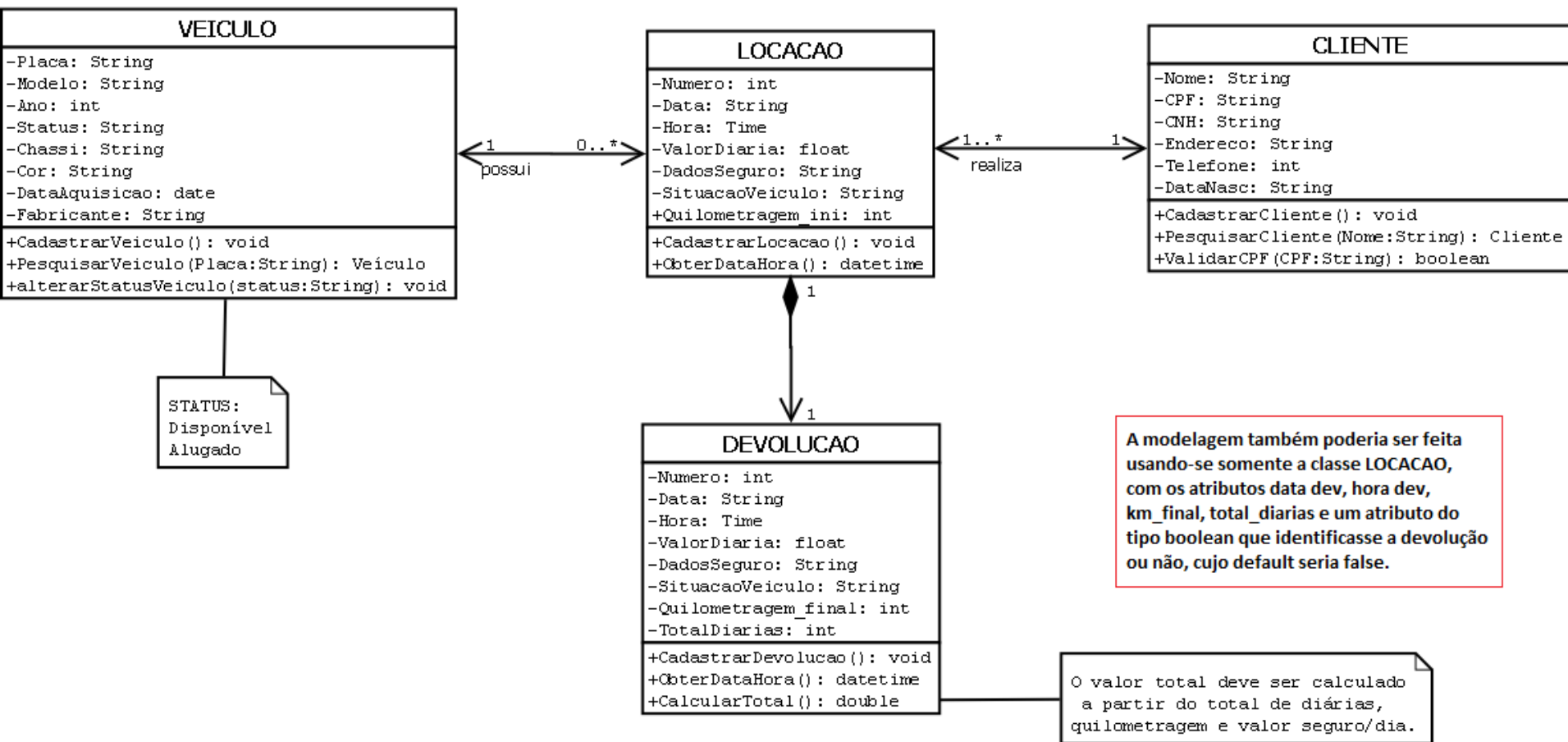


# DIAGRAMA DE CASOS DE USO



# DIAGRAMA DE CLASSES - RESPOSTA

## SISTEMA DE LOCAÇÃO DE VEÍCULOS



# REFERÊNCIAS

- GUEDES, Gilleanes T. A. **UML 2: uma abordagem prática**. 2. ed. São Paulo: Novatec Editora, 2011.