

# Projeto de Sistemas Orientado a Objetos

**UNIP**

UNIVERSIDADE PAULISTA



# Especificação do Software

A especificação envolve a definição clara e precisa dos requisitos do sistema a ser desenvolvido.

## **Atividades Incluídas:**

- **Levantamento de Requisitos:** Entendimento das necessidades dos usuários e stakeholders.
- **Documentação:** Criação de documentos que descrevem o comportamento esperado do sistema.

**Exemplo:** Para um sistema de gestão de tráfego em uma cidade inteligente, os requisitos podem incluir a capacidade de monitorar e controlar semáforos em tempo real.

# Especificação do Software

A gestão de requisitos é um conjunto de atividades que tem como principal objetivo ajudar a equipe de projeto a **identificar, controlar e rastrear requisitos e modificações de requisitos em qualquer época, à medida que o projeto prossegue.**

São atribuições na etapa de especificação de requisitos:

- I. Identificar as expectativas e necessidades dos stakeholders com relação ao software a ser desenvolvido.
- II. Distribuir os requisitos em categorias, explorar as relações entre eles e classificar sua importância para os stakeholders.
- III. Produzir um documento de especificação de requisitos, de forma que todos os stakeholders possam entendê-lo.
- IV. Examinar a especificação do software para assegurar que todos os requisitos foram definidos sem inconsistências.



# Documentação

12 de fev. de 2025 | Módulo Empenho

Participantes: ;

## Anotações

Na reunião foram tratadas as solicitações feitas no chamado 7843 (<https://?id=7843>). Em relação a melhoria no Módulo empenho, foram acordados:

### 1. Desenvolvimento de Botão para Cancelamento:

- Criação de um botão para **cancelamento total ou parcial** do empenho, mantendo-o disponível para histórico.
- No cancelamento, será necessário informar o **número do processo eletrônico** criado para o cancelamento.

### 2. Opção de Cancelamento Parcial e Total:

- No caso do cancelamento parcial, será possível informar manualmente o valor a ser cancelado. O **valor informado pode ser inferior ao valor residual ou corresponder ao valor real residual**. A inclusão automática do valor residual não poderá ser implementada, pois existem implicações caso isso ocorra. Deverá ser implementada uma validação para garantir que o cancelamento não ultrapasse o valor residual.
- Caso o cancelamento total seja selecionado, o saldo total do empenho será dissolvido automaticamente.

### 3. Validação para Cancelamento Parcial:

- Será implementada uma validação para garantir que o **cancelamento parcial não ultrapasse o valor residual**.

### 4. Inclusão de Opções no Filtro "Situação":

- O filtro "Situação" incluirá as opções **"Cancelado Parcialmente"** e **"Cancelamento Total"**, além das opções já existentes (concluído, não concluído, iniciado e não iniciado).

## [almoxarifado] Implementação de Botão de Cancelamento de Empenho

Edit

Open Issue created 1 week ago by

### Origem

- Data da Conferência: 12/02/2025
- Plataforma: Google Meet
- Participantes: ,
- Chamado Relacionado: 7843 (<https://?id=7843>)
- Ata: [Notas\\_-\\_Módulo\\_Empenho](#)

### Descrição

Perfil de teste:

A solicitação é para a criação de um botão de cancelamento com opções de cancelamento total ou parcial do empenho, além de ajustes no filtro de busca e na tabela, especificamente na coluna Situação" e "Nota sobre o prazo".

#### 1. Desenvolvimento de Botão para Cancelamento:

- Criação de um botão para cancelamento com as opções de total ou parcial do empenho
- Criação de um botão para cancelamento total ou parcial do empenho, mantendo-o disponível para histórico.
- No cancelamento, será necessário informar o número do processo eletrônico criado para o cancelamento.

#### 2. Opção de Cancelamento Parcial e Total:

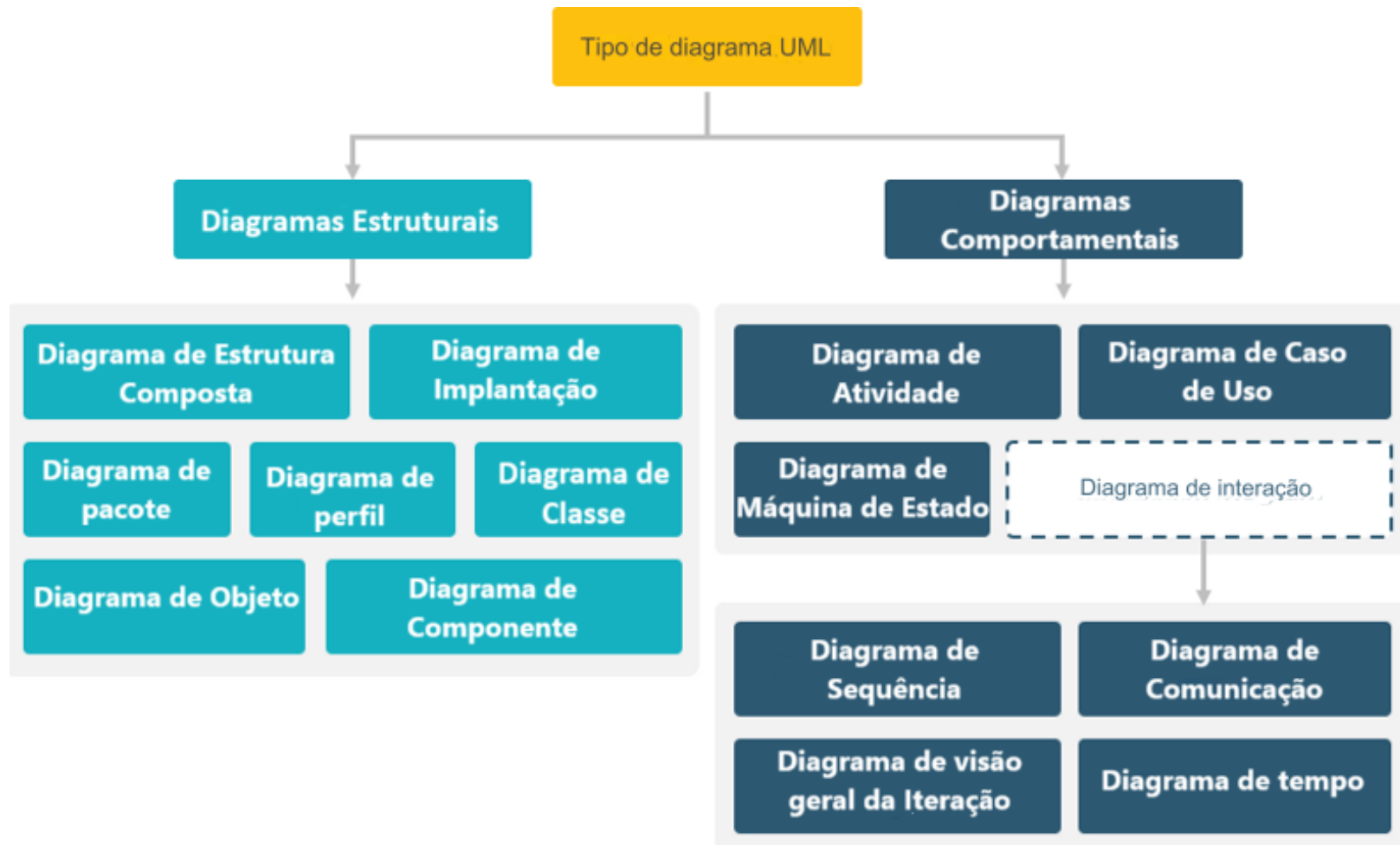
- No caso do cancelamento parcial, será possível informar manualmente o valor a ser cancelado. O valor informado pode ser inferior ao valor residual ou corresponder ao valor real residual. A inclusão automática do valor residual não poderá ser implementada, pois

# Unified Modeling Language (UML)

- A Linguagem de Modelagem Unificada é uma linguagem de modelagem padronizada e de uso geral que agora é gerenciada pelo Object Management Group (OMG) como um padrão de fato da indústria.
- A UML inclui um conjunto de técnicas de notação gráfica para a criação de modelos visuais de sistemas com uso intensivo de software.

# Unified Modeling Language (UML)

- Existem 14 tipos de diagramas UML divididos em duas categorias:



# Unified Modeling Language (UML)

Grady Booch, um dos mais importantes desenvolvedores de Linguagem de Modelagem Unificada, disse que:

**“80% do software requer apenas 20% de UML”**

# Diagrama de Casos de Uso

São a principal forma de requisitos de sistema/software para um novo programa de software não desenvolvido. Os casos de uso especificam o comportamento esperado (o quê), em vez do método exato para alcançá-lo (como).

Uma vez especificados os casos de uso, eles podem ser representados usando representações textuais e visuais (ou seja, diagramas de casos de uso).

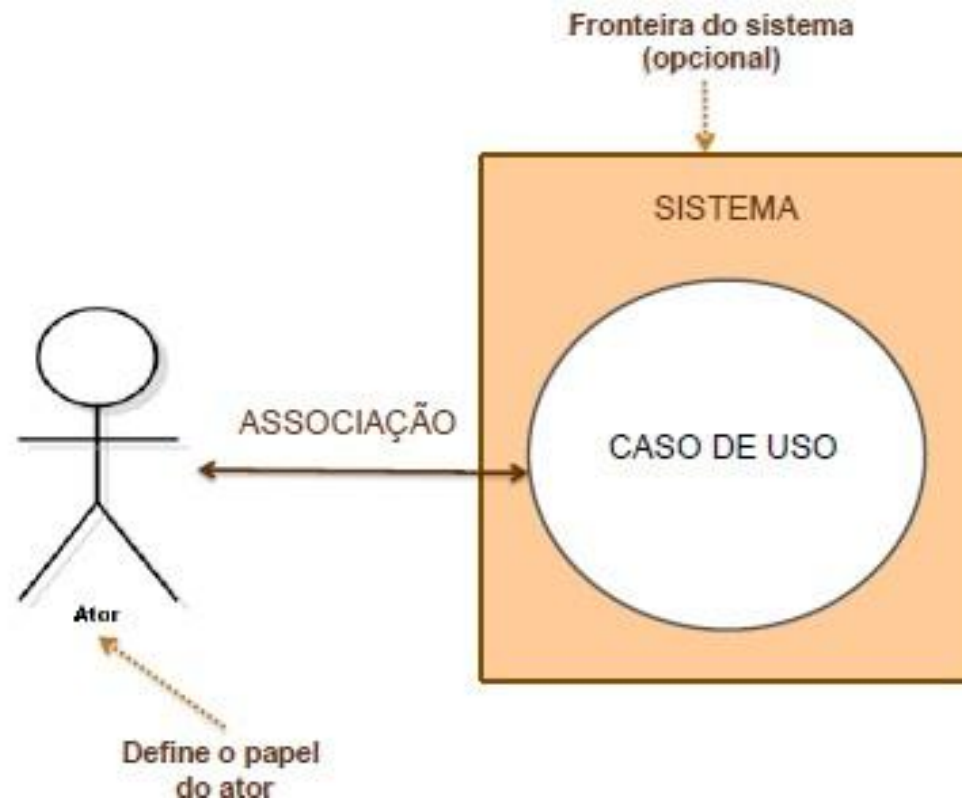
Um conceito chave da modelagem de casos de uso é que ela nos ajuda a projetar o sistema da perspectiva do usuário final. É uma técnica eficaz para comunicar o comportamento do sistema nos termos do usuário, especificando todo o comportamento do sistema visível externamente.



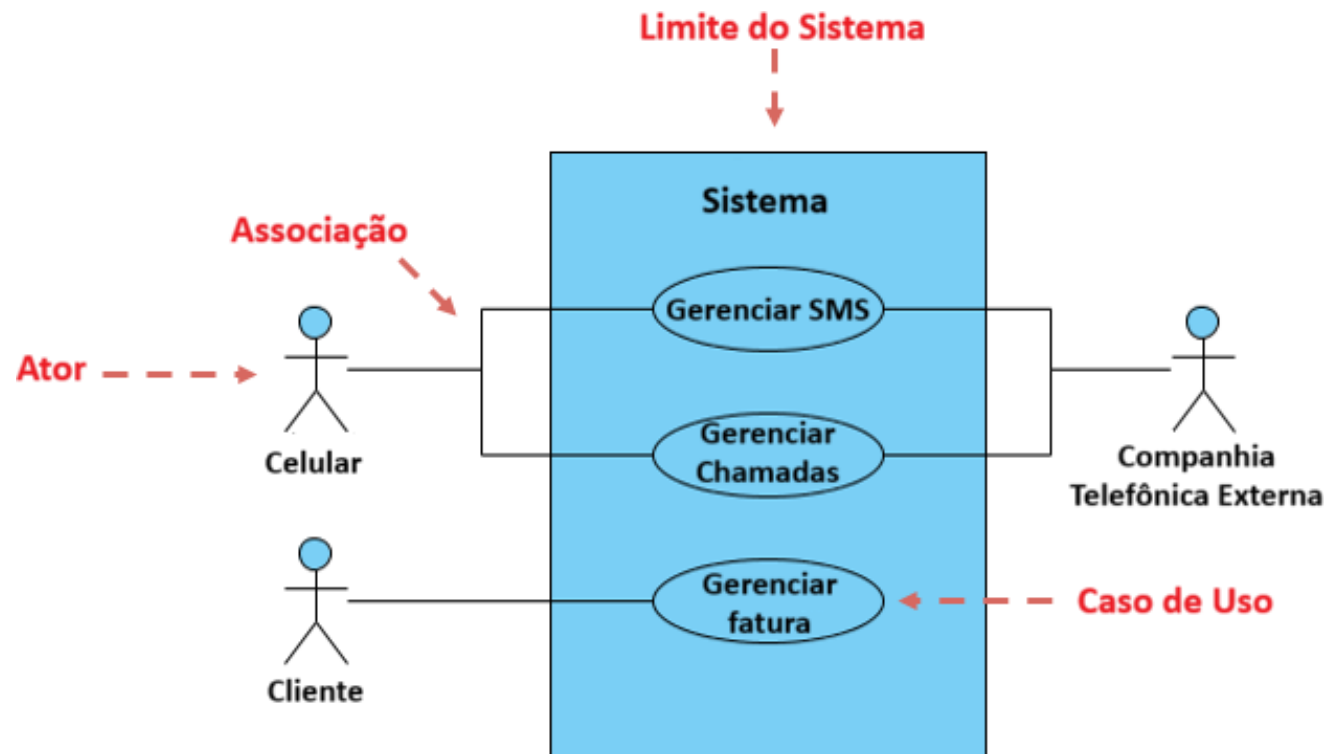
# Diagrama de Casos de Uso

- É o diagrama mais geral e informal da UML
- Objetiva identificar os atores e as funcionalidades do sistema.
- Pode ser apresentado aos usuários para que tenham uma visão geral de como o sistema funcionará.
- É consultado durante todo o processo de desenvolvimento de um sistema.
- É usado como base para outros diagramas.

# Diagrama de Casos de Uso – Elementos básicos



# Diagrama de Casos de Uso



# Diagrama de Casos de Uso - Atores

- Representam os papéis desempenhados pelos diversos usuários que poderão utilizar ou interagir com os serviços e funções do sistema.
- Pode ser qualquer elemento externo que interaja com o sistema, inclusive um software ou hardware.
- Exemplos típicos: cliente, aluno, supervisor, professor, impressora fiscal, dispositivo de conexão de rede etc.
- Identificando atores de um sistema:
  - Quem utilizará a principal funcionalidade do sistema?
  - Quem (ou o que) tem interesse nos resultados do sistema?
  - Quais dispositivos (hardware) são necessários?
  - Com quais outros sistemas o sistema em foco irá interagir?



# Diagrama de Casos de Uso – Caso de Uso

- Referem-se aos serviços, tarefas ou funções que podem ser utilizados pelos usuários do sistema;
- São usados para expressar e documentar os comportamentos das funções do sistema;
- Em geral, podemos associar um caso de uso a uma tela (ou página) de um sistema, apesar de isto não ser uma regra;
- Contém um texto descrevendo o serviço (iniciando-se com um verbo);
- Exemplos:
  - Cadastrar produto;
  - Gerar relatório de vendas;
  - Emitir NF-e.





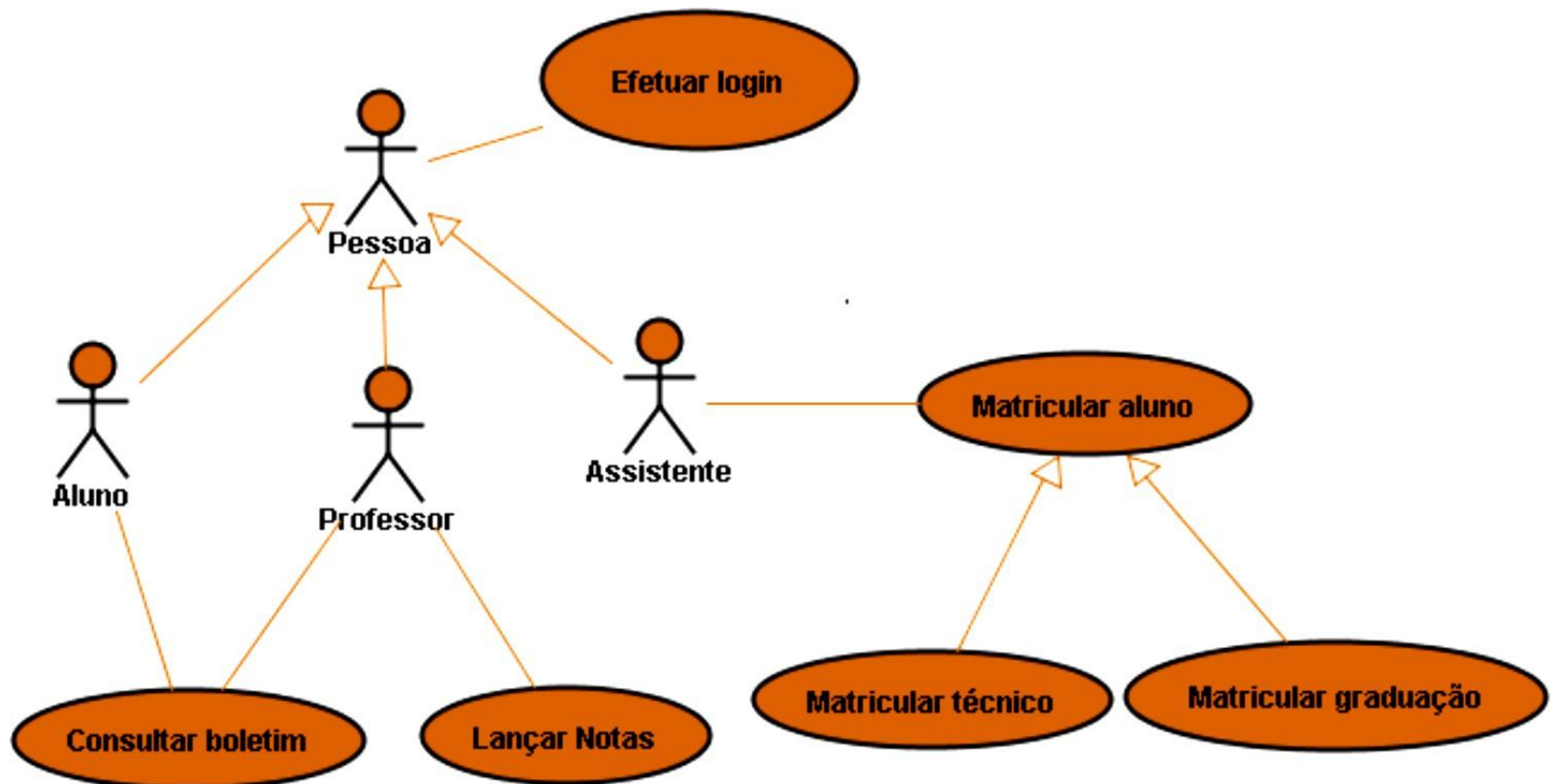
# Diagrama de Casos de Uso – Associações

São representadas por uma linha que liga o ator ao caso de uso:



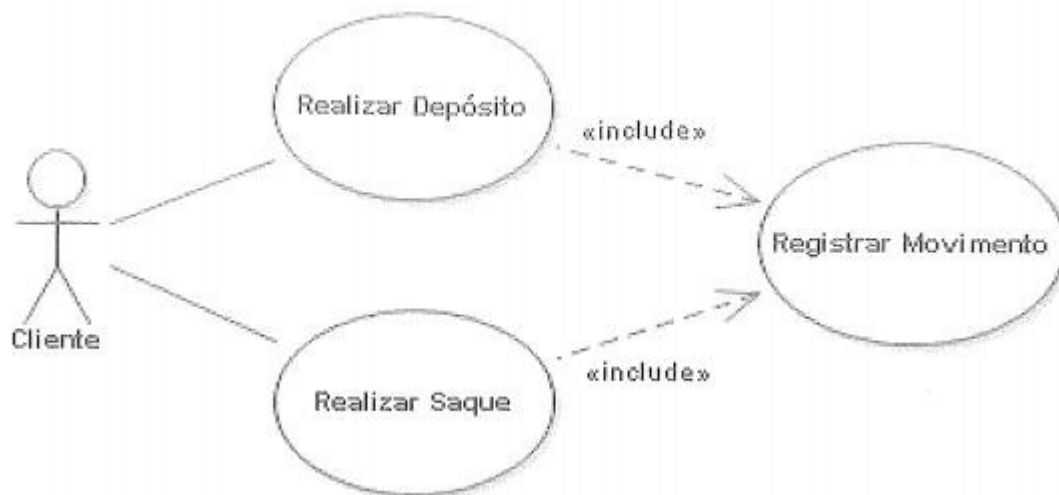
# Generalização e Especialização

- Forma de associação na qual existem dois ou mais casos de uso com características semelhantes;
- Também é possível com atores;



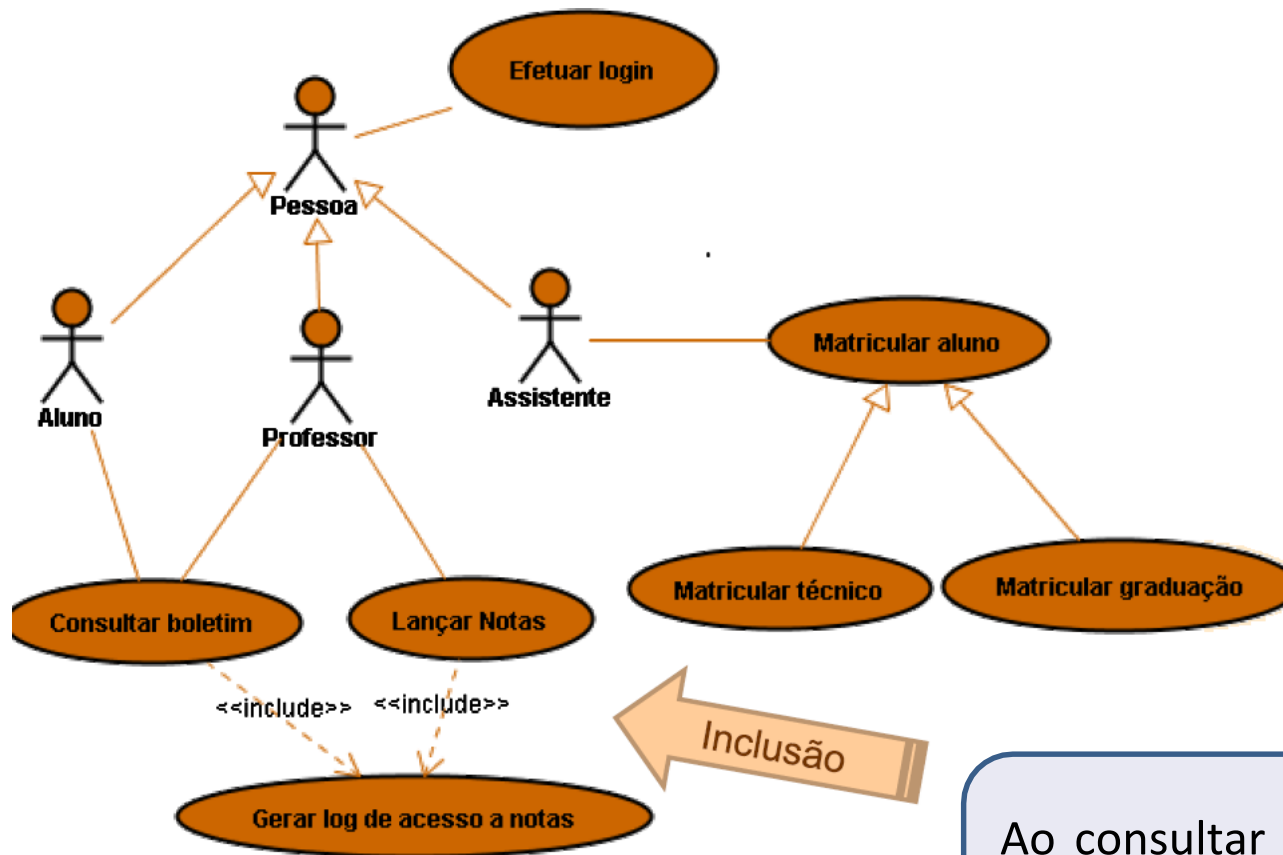
# Relacionamentos - Inclusão

- A execução de um caso de uso obriga a execução de um outro;
- Pode ser comparado à chamada de uma sub-rotina;
- Representada por uma reta tracejada com uma seta apontando para o caso de uso que foi incluído;
- Estereótipo com o texto `<<include>>`.
- Esse tipo de relacionamento indica **OBRIGATORIEDADE**: um Caso de Uso de inclui outro executa os dois!



Ao realizar tanto um saque quanto um depósito, a operação será registrada

# Relacionamentos - Inclusão



Ao consultar boletim e lançar notas, será gerado um log de acesso às notas.

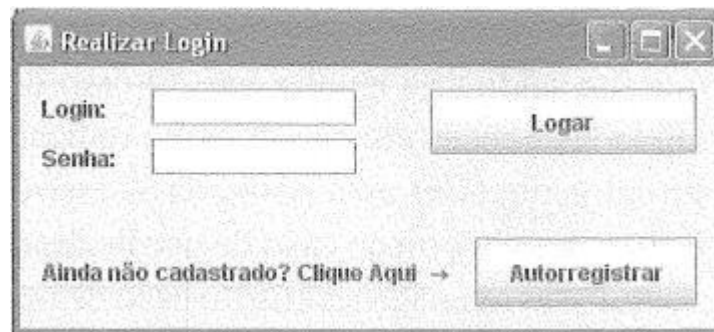
## Relacionamentos - Extensão

- É utilizada para descrever cenários **OPCIONAIS** de um Caso de Uso.
- Nesse caso existe um teste para determinar se o Caso de Uso extendido é executado, não é obrigatório como no <<include>>
- Um Caso de Uso pode ter vários outros como extensões que só serão executados em determinadas situações.
- Só ocorrerá se uma determinada condição for satisfeita;
- Representada por uma reta tracejada com uma seta apontando para o caso de uso que recebe a extensão;
- Vejamos um exemplo com LOGIN:



# Relacionamentos - Extensão

- Tela de Login:

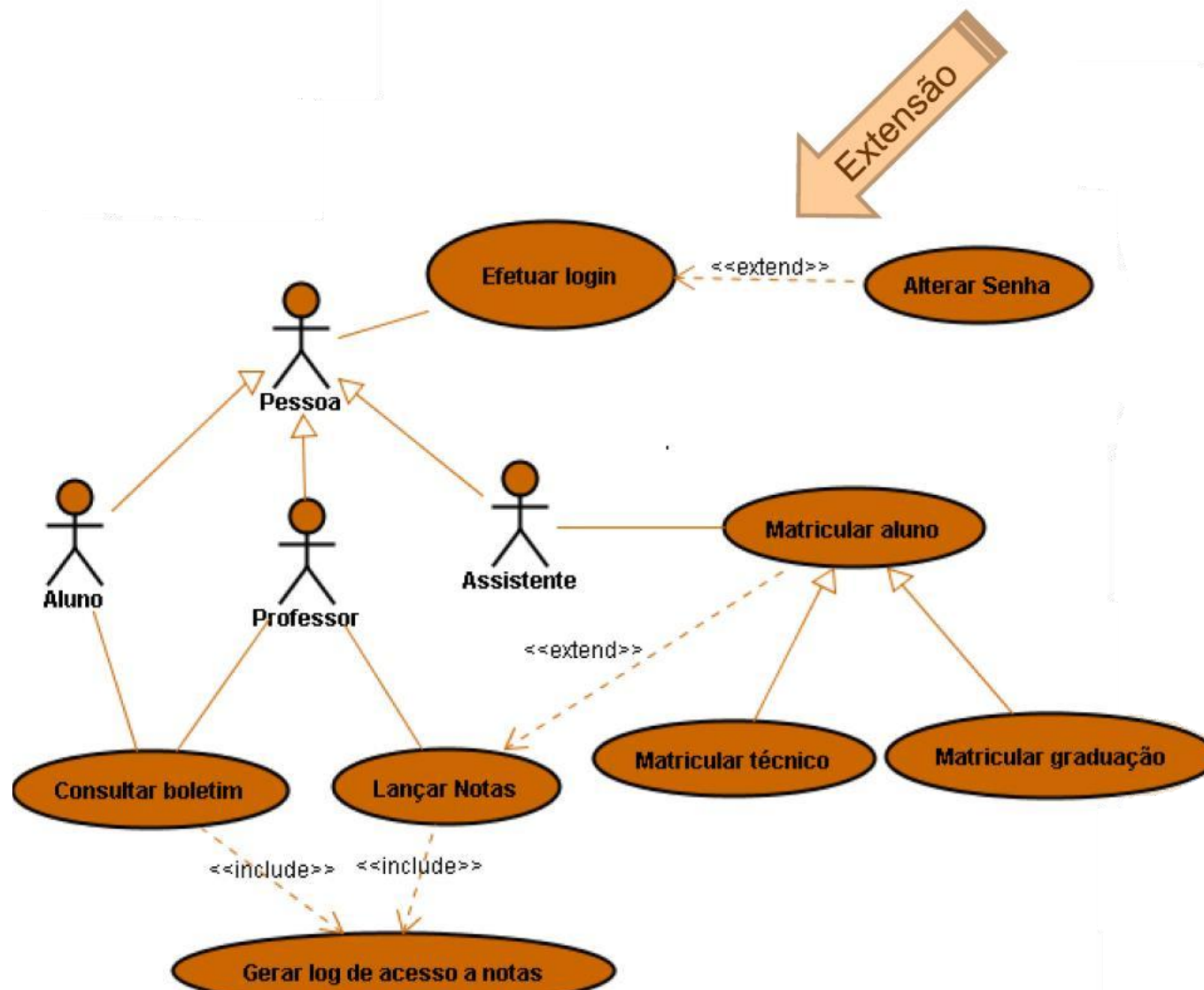


A screenshot of a web application window titled "Realizar Login". It contains two input fields labeled "Login:" and "Senha:". To the right of the "Senha:" field is a button labeled "Logar". Below the input fields, there is a text label "Ainda não cadastrado? Clique Aqui →" and a button labeled "Autorregistrar".

- Caso o usuário não tenha conta ele pode se registrar:



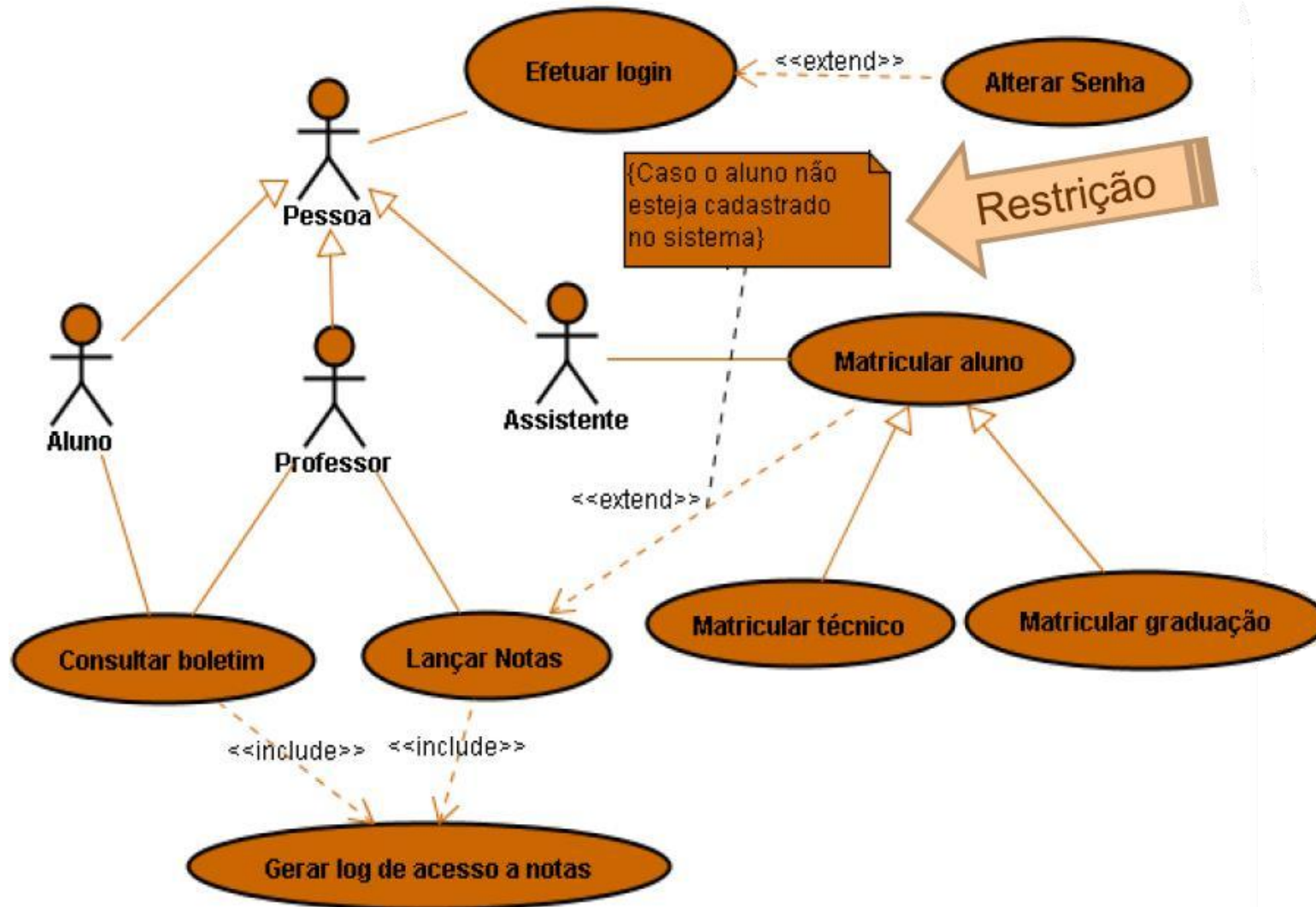
# Relacionamentos - Extensão



# Relacionamentos - Restrições

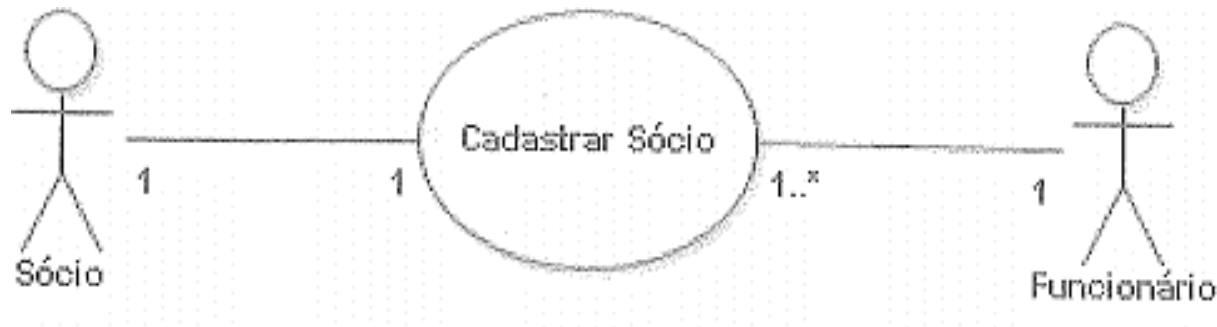
- Às vezes, não fica claro a condição que deve ser satisfeita para que um caso de uso seja executado;
- Nesses casos, podemos usar restrições com uma nota explicativa determinando a condição para que o caso de uso seja executado;
- As restrições são compostas por um texto entre chaves:

# Relacionamentos - Restrições



# Multiplicidade

- Especifica o número de vezes que um Ator pode utilizar um Caso de Uso:



- Nesse caso o Sócio só pode se cadastrar uma vez, enquanto que o Funcionário pode cadastrar diversos Sócios.
- Também é possível ver que apenas um Sócio e um Funcionário utilizam a função.



# Documentação de Casos de Uso

- Descrição bastante simples do caso de uso;
- Tem como objetivo informar os atores que interagem com o sistema e as etapas que devem ser executadas pelo ator e pelo sistema para que o caso de uso execute sua função;
- Não há um modelo padrão para especificação de caso de uso;
- Recomenda-se que seja simples e de fácil acesso.

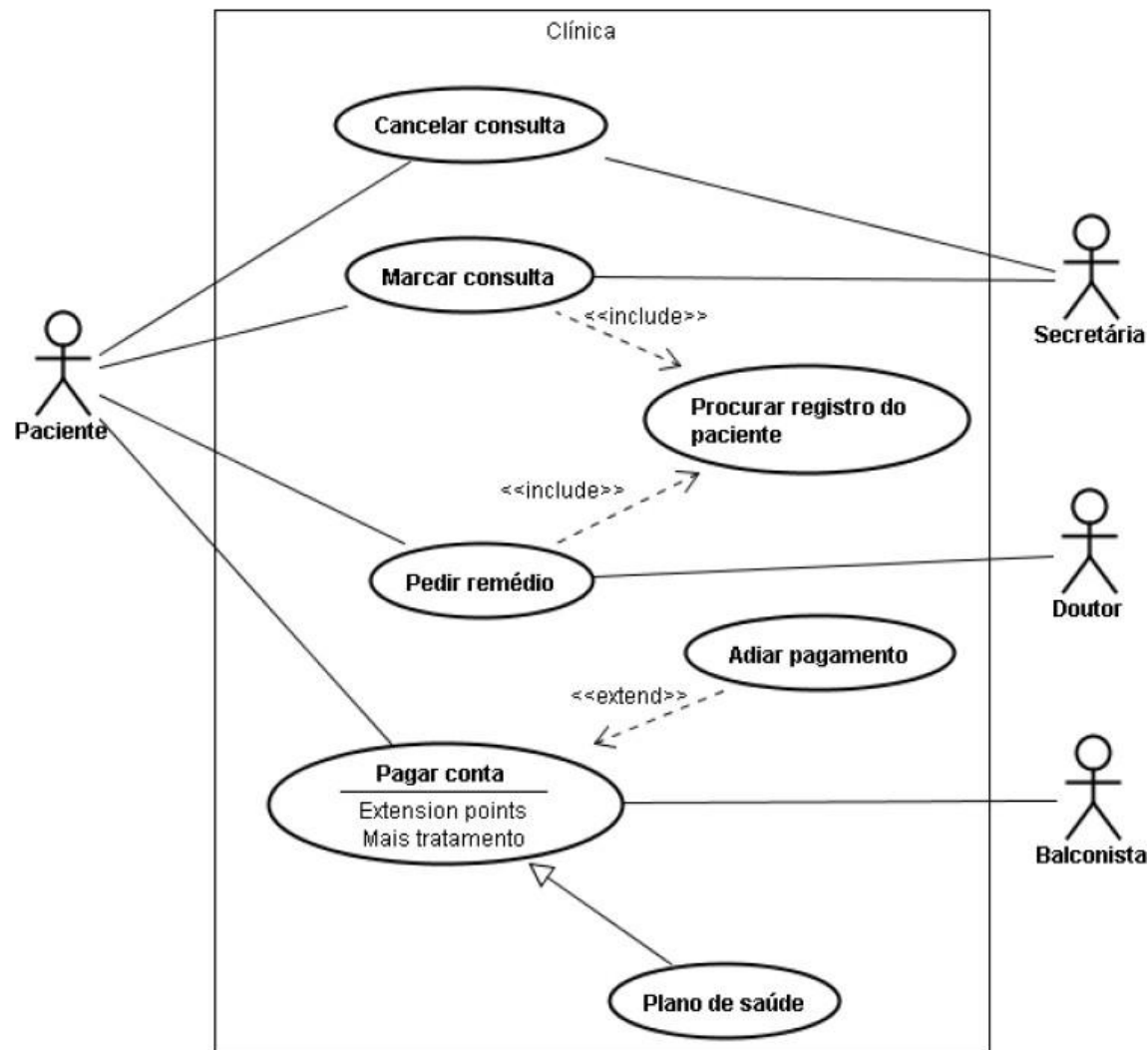
# Exemplo de Documentação

Nome do Caso de Uso	Abrir Conta
Caso de Uso Geral	
Ator Principal	Cliente
Atores Secundários	Funcionário
Resumo	Este caso de uso descreve as etapas percorridas por um cliente para abrir uma conta corrente
Pré-condições	O pedido precisa ser aprovado
Pós-condições	É necessário realizar um depósito inicial
Ações do Ator	Ações do Sistema
1. Solicitar conta	
	2. Consultar cliente por seu CPF
	3. Avaliar pedido do cliente

# Exemplo de Documentação

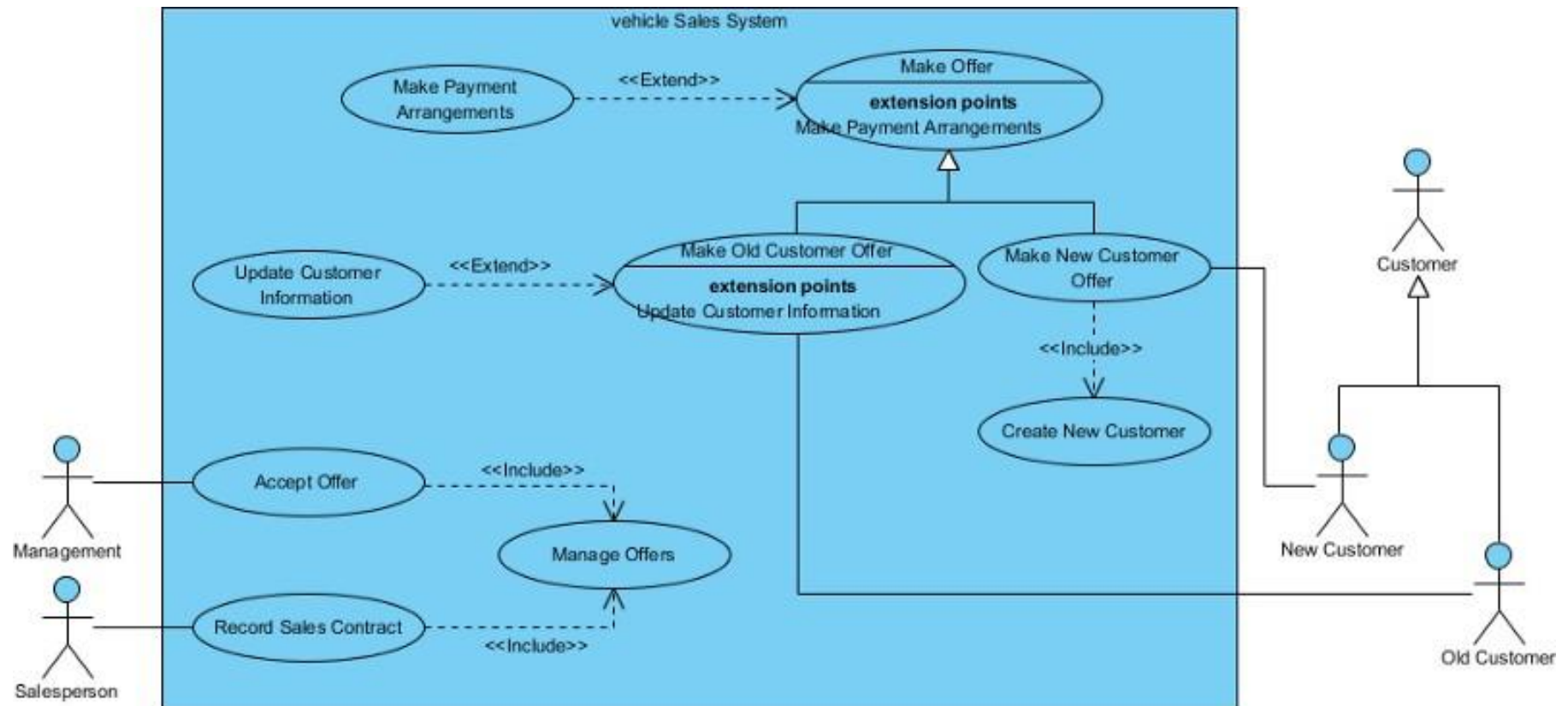
	4. Aprovar pedido
5. Escolher a senha da conta	
	6. Abrir conta
	7. Definir cliente como ativo
8. Fornecer valor a ser depositado	
	9. Emitir cartão da conta
Restrições/Validações	1. Para abrir uma conta é preciso ser maior de idade
	2. O valor mínimo de depósito é R\$ 5,00

# Exemplo de Diagrama de Caso de Uso

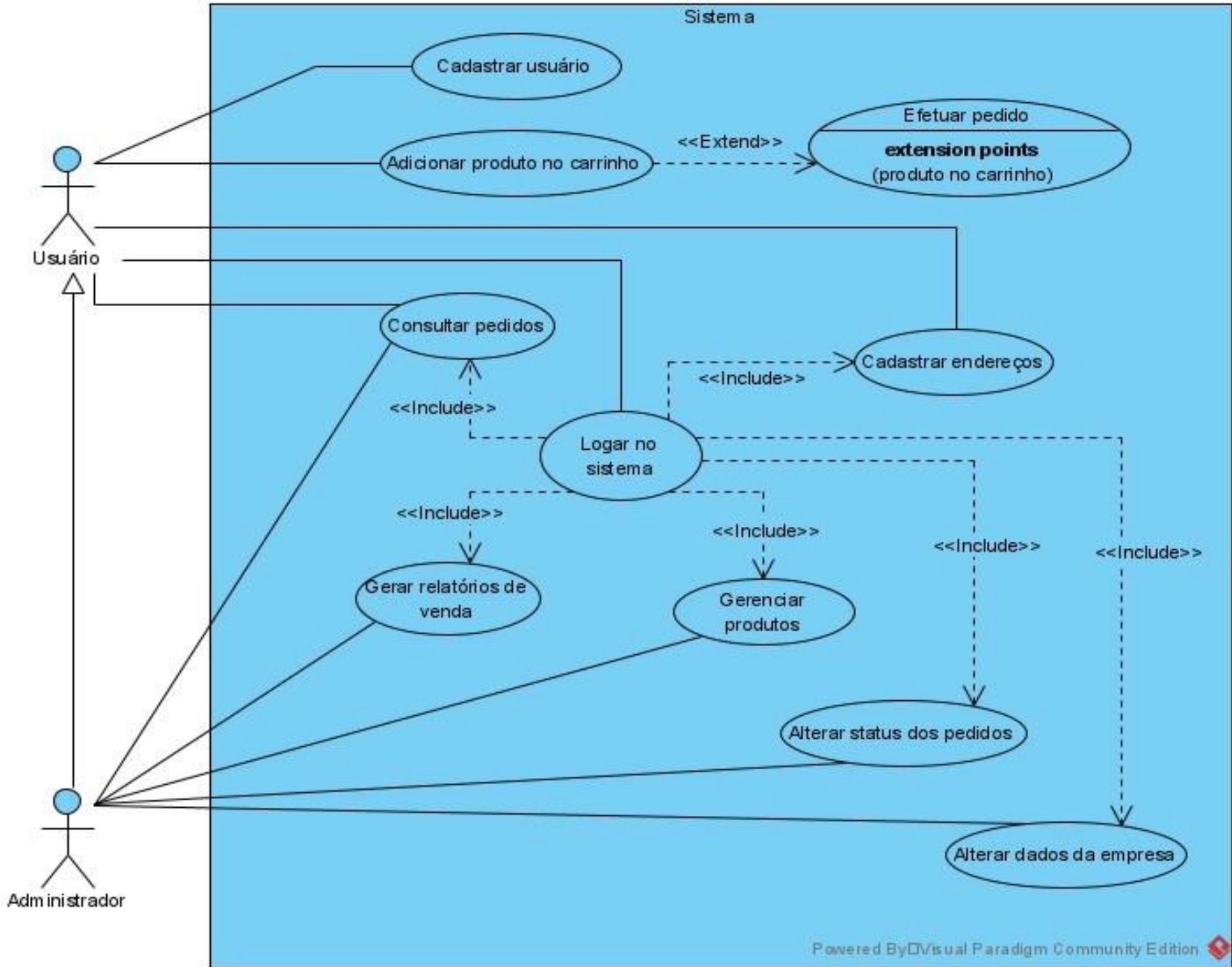


# Diagrama de Casos de Uso

A figura mostra um exemplo de diagrama de caso de uso para um sistema de veículo. Como você pode ver, mesmo um sistema tão grande como um sistema de vendas de automóveis não contém mais do que 10 casos de uso!







# Requisitos Casos de Uso

**Caso de uso:** Cadastrar Usuário

**Atores:** Usuários e Administrador

**Precondições:** nome de usuario, senha, email, celular e imagem.

**Pós-condições:** dados do usuário inseridos no banco de dados, usuário pode realizar login no sistema

**Sequência típica de eventos (Fluxo Principal):**

Esse caso de uso inicial quando:

1. [IN] O ator seleciona a opção cadastrar-se na tela inicial
2. [OUT] O sistema apresenta a tela de cadastro
3. [IN] O usuário informa os dados desejados
4. [OUT] O sistema verifica os dados informados

**Caso de uso:** Logar no sistema

**Atores:** Usuários e Administrador

**Precondições:** Usuário cadastrado, digitar usuário e senha corretos

**Pós-condições:** Ator logado no sistema

# Requisitos Casos de Uso

## **Sequência típica de eventos (Fluxo Principal):**

Esse caso de uso inicial quando:

1. [IN] O ator informa o usuario e senha.
2. [OUT] O sistema verifica se usuário e senha estão cadastrados.
3. [OUT] Ator tem acesso a tela de usuário

## **Tratamento de Exceções e Variantes:**

Exceção 1a: Nome de usuário ou senha não encontrados.

- 1.1 O sistema informa ao ator que os dados não conferem.
1. 1.2 O sistema disponibiliza opção de alteração de senha ao ator
2. 1.3 Retorna a tela de login.

# Requisitos Casos de Uso

**Caso de uso:** Adicionar produto no carrinho de compras

**Atores:** Usuários

**Precondições:** produto com status “Disponível”.

**Pós-condições:** Produto adicionado ao carrinho de compras.

**Sequência típica de eventos (Fluxo Principal):**

Esse caso de uso inicia quando:

1. [OUT] O sistema verifica o status do produto.

Variante 1.1: Produto indisponível

- 1.1.1 [OUT] O sistema não lista o produto

Variante 1.2: Produto disponível

- 1.1.2 [OUT] O sistema apresenta o produto para seleção

- 1.1.3 [IN] O cliente seleciona o produto que deseja adicionar ao carrinho

- 1.1.4 [OUT] O sistema apresenta uma nova página contendo os detalhes dos produtos e o campo “Quantidade”

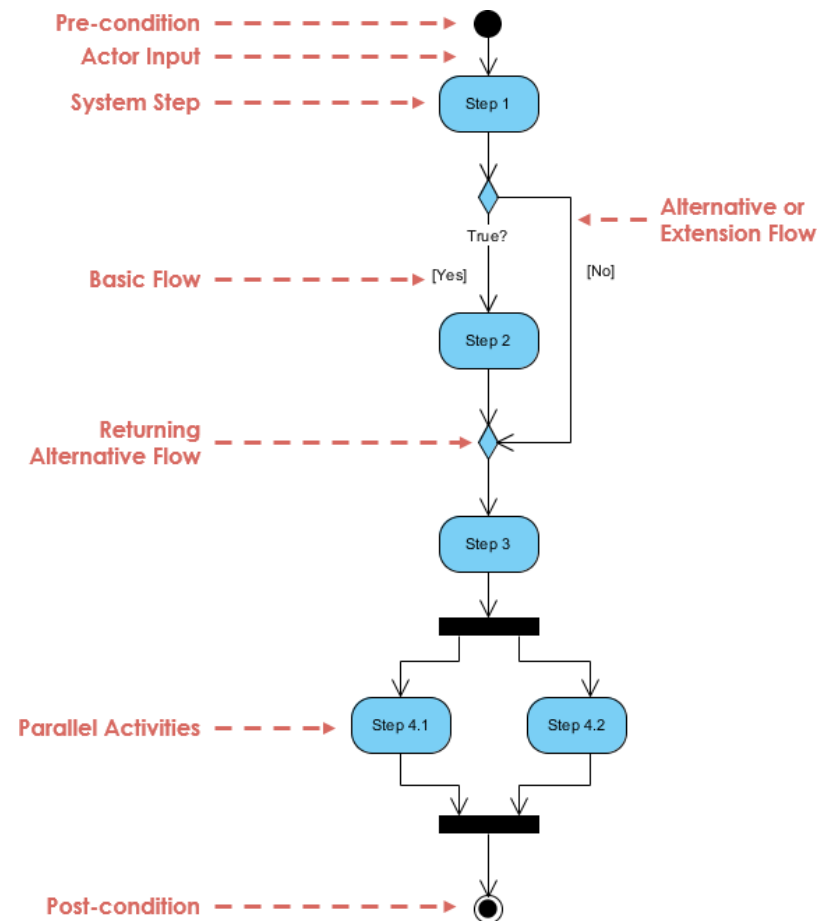
- 1.1.5 [IN] O cliente insere a quantidade desejada e clica em “Adicionar ao carrinho”

- 1.1.6 [OUT] O sistema calcula o valor do pedido, adiciona o produto ao carrinho e solicita ao cliente se deseja finalizar a compra ou “continuar comprando”.

# Diagrama de Atividade

O diagrama de atividades é outro diagrama **comportamental** importante entre os diagramas UML, usado para descrever os **aspectos dinâmicos** do sistema.

Os diagramas de atividades são essencialmente uma versão avançada de fluxogramas, que **modelam o fluxo de uma atividade para outra**.



# Diagrama de Atividade

- Descrevem como as atividades são coordenadas para fornecer serviços que podem estar em diferentes níveis de abstração.
- Também é adequado para modelar como coleções de casos de uso se coordenam para representar fluxos de trabalho de negócios.

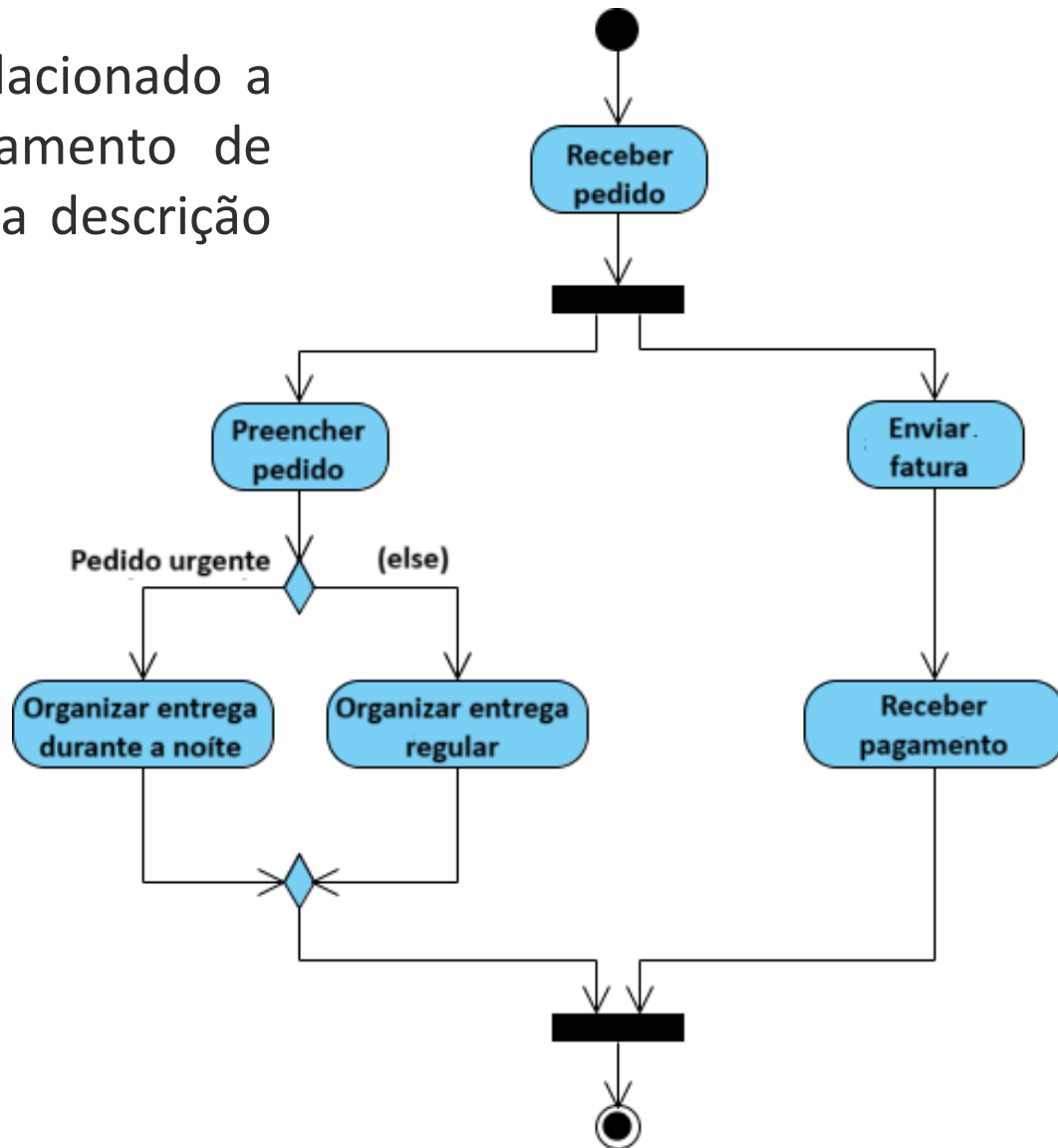
## Desvantagens

- Não definem quais objetos executam as funcionalidades (Diagrama de Estado e Seqüencia)
- Nem o fluxo de mensagens (Diagrama de Seqüência)

# Ordem de Processo – Descrição do Problema

Dada a descrição de um problema relacionado a um fluxo de trabalho para processamento de pedidos, vamos modelar visualmente a descrição usando um diagrama de atividades:

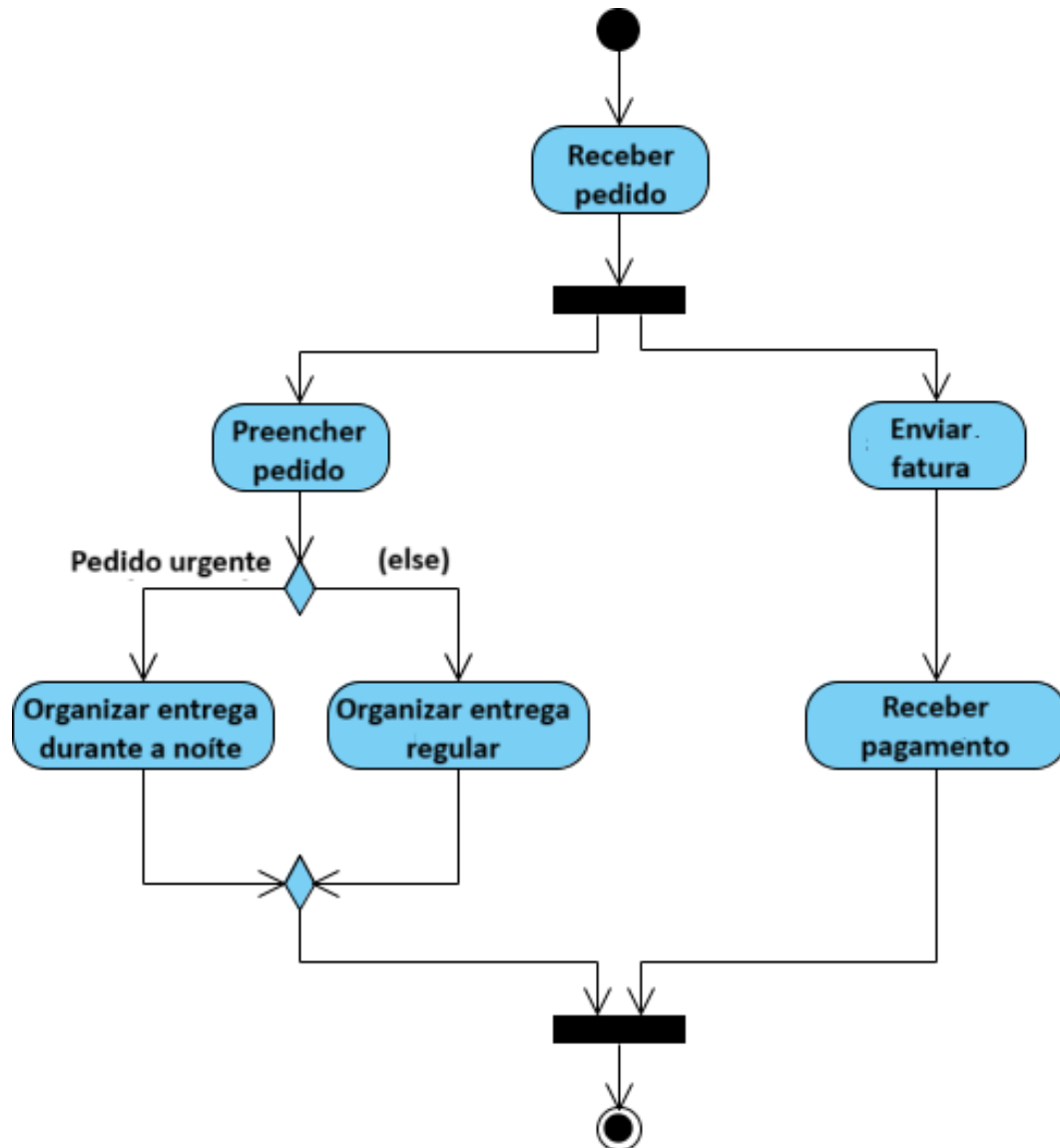
- Assim que um **pedido é recebido**, a atividade é dividida em dois conjuntos paralelos de atividades. Uma parte **atende e envia o pedido** e a outra **cuida do faturamento**.





# Ordem de Processo – Descrição do Problema

- Quando se trata de atender um pedido, o método de entrega é determinado condicionalmente. **Execute atividades de entrega noturna ou atividades de entrega regulares** de acordo com a situação.
- Finalmente, atividades paralelas se combinam para fechar o pedido.





# Classificação

- Quando éramos crianças, os adultos nos ensinaram a pensar de forma orientada a objetos;
- Por exemplo, pensávamos em conceitos simples como pessoa, carro, mala e coelho;
- Quando as pessoas pensam assim (sejam elas crianças ou não), são definidas classes, ou seja, um conjunto de objetos;
- O nosso aprendizado é obtido por meio da classificação, isto é, formar grupos de objetos com características e comportamentos semelhantes.



# Classe

Uma **classe** representa a descrição de um **objeto do mundo real**. Elas devem ser retiradas do domínio do problema e nomeadas de acordo com o que representam no sistema.

# Abstração

- A abstração é essencial para identificarmos classes;
- Consiste na seleção de alguns aspectos de domínio do problema a modelar, desconsiderando os irrelevantes para o nível de abstração em questão;

## Mundo REAL



CARRO DE SOM



JOÃO



MAMADEIRA

- Indispensável na modelagem de objetos reais porque, no mundo real, quase tudo é complexo:

## Problema MODELADO



# Objeto



# Objeto

- É uma entidade real ou abstrata, que modela um conceito presente na realidade humana, ocupando espaço físico ou lógico;
- Um objeto é uma pessoa, um lugar, é a base para todos os outros conceitos da orientação a objetos;
- Facilita a compreensão do mundo real e oferece uma base real para implementação em computador;

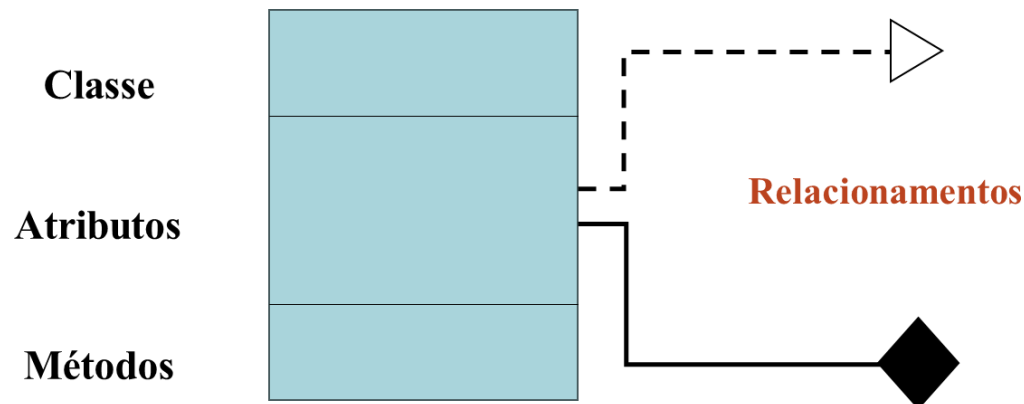
Um objeto denota uma entidade de natureza física, conceitual ou de software:

1. Entidades físicas: um carro, uma pessoa, um livro;
2. Entidade conceitual: um DER de uma sistema;
3. Entidade de software: um *radiobutton* em uma página web.

# Diagrama de Classe

O diagrama de classes é o mais utilizado e, portanto, será discutido primeiro.

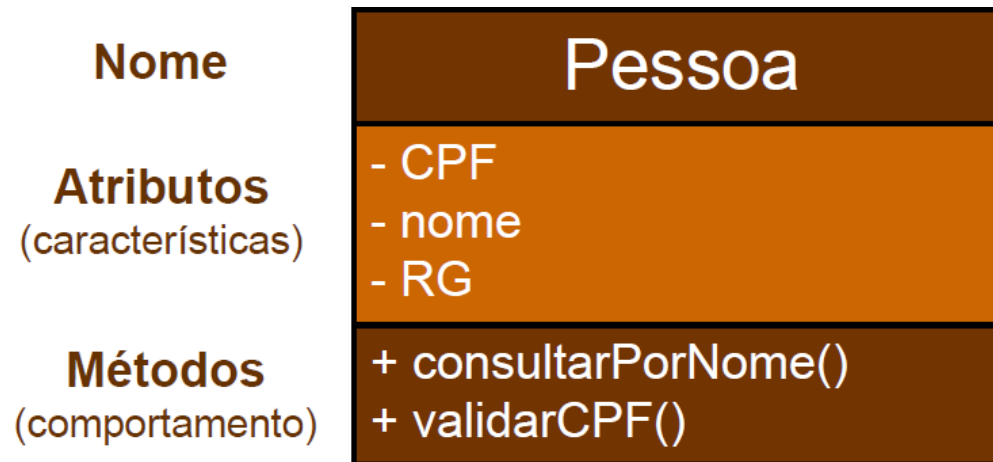
Na engenharia de software, um diagrama de classes em UML é um **diagrama de estrutura estática** que descreve a estrutura de um sistema mostrando suas **classes**, suas **propriedades**, **operações** (ou métodos) e os **relacionamentos entre objetos**.



# Diagrama de Classe

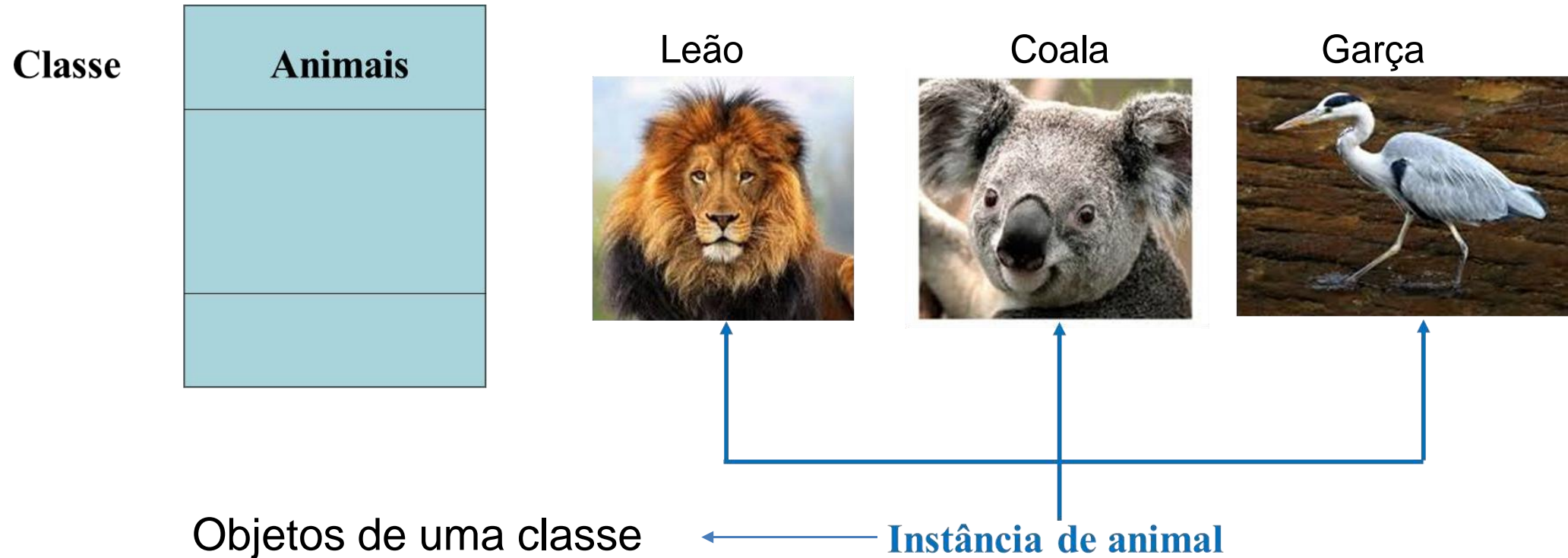
Representada por um retângulo que pode possuir até três divisões:

- Nome da classe
- Atributos pertencentes à classe
- Possíveis métodos da classe



# Sistema Zoológico - Classe

Vamos descrever as coisas que podem ser encontradas no sistema do zoológico, em que representamos essas coisas por classes.





# Sistema Zoológico - Atributos

É um pedaço significativo de dados que contém valores que descrevem cada instância dessa classe. Também são conhecidos como campos, variáveis ou propriedades.

Para nossa classe animal podemos criar atributos como:

- ✓ nome
- ✓ núm.ident
- ✓ idade

Classe	Animais
Atributos	<b>-nome: string</b> <b>-núm.ident: int</b> <b>-idade: int</b>

**nome: Vini**  
**núm.ident: 304**  
**idade: 35**



# Sistema Zoológico - Métodos

Também conhecidos como operações ou funções, permitem especificar as características **comportamentais** de uma classe.

Classe	Animais
Atributos	-nome: string -núm.ident: int -idade: int
Métodos	-definirNome() -comer()

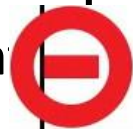
nome: Vini Theo  
núm.ident: 304  
idade: 35



# Sistema Zoológico - Visibilidade

A visibilidade define a acessibilidade para esse atributo ou método

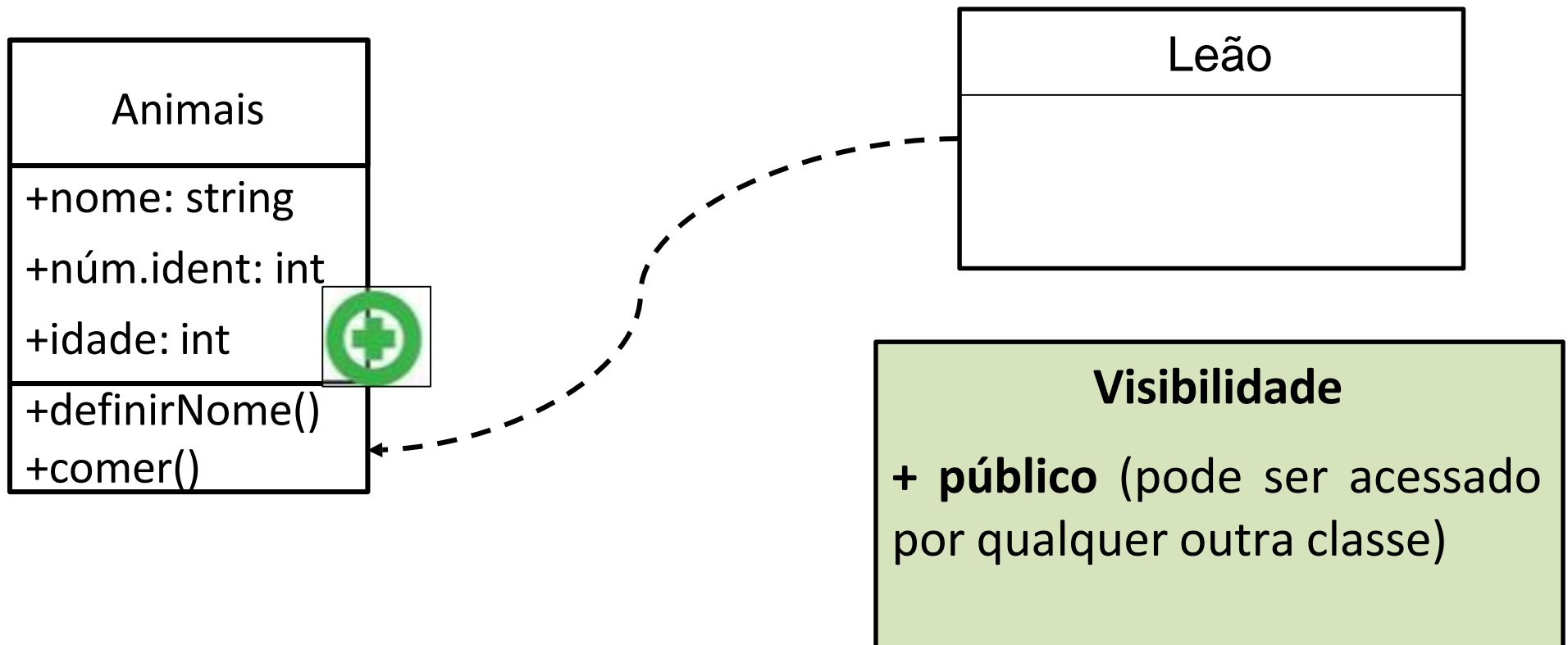
Animais	
-nome:	string
-número:	m.ident: int
-idade:	int
-definirNome():	
-comer():	



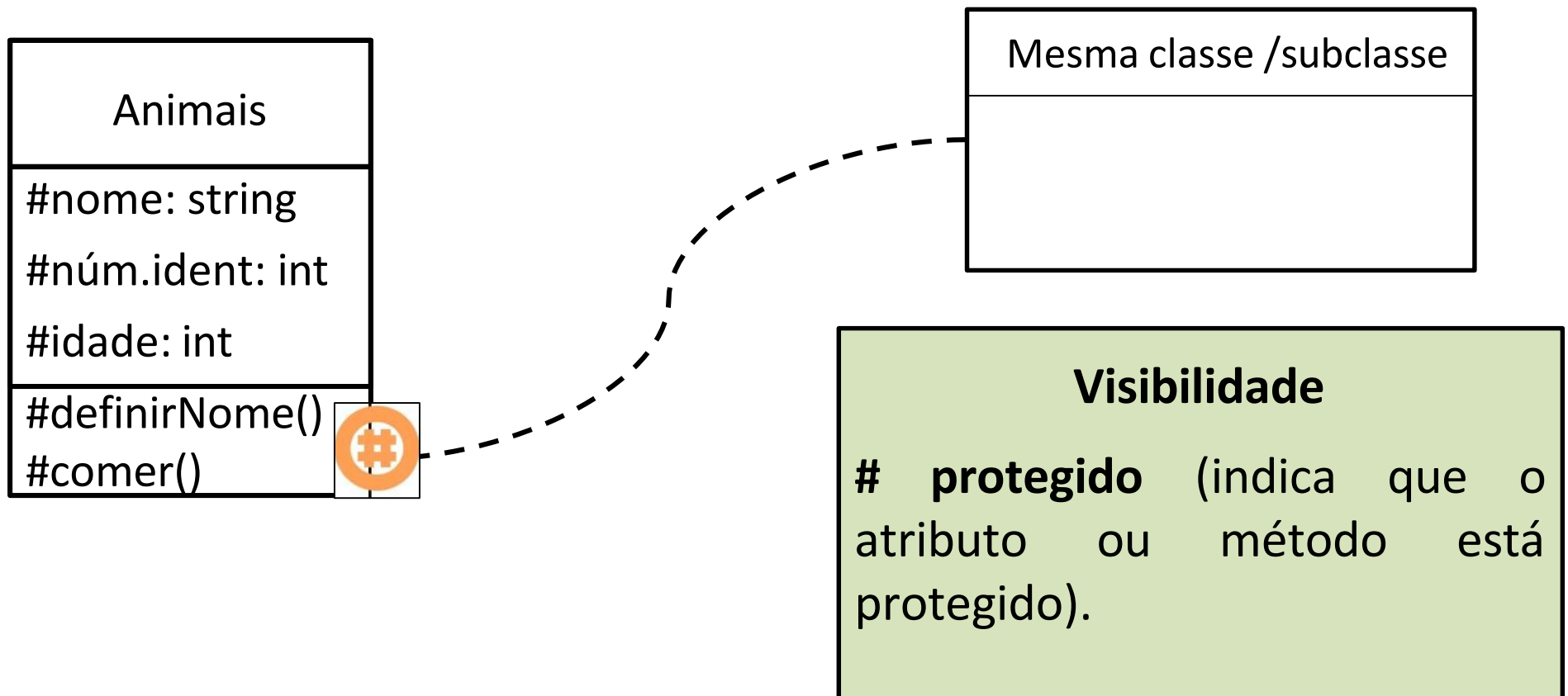
Leão

Visibilidade
- <b>privado</b> (não podem ser acessados por qualquer outra classe ou subclasse)

# Sistema Zoológico - Visibilidade



# Sistema Zoológico - Visibilidade

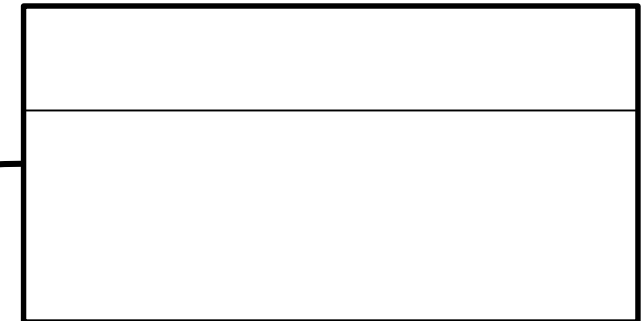
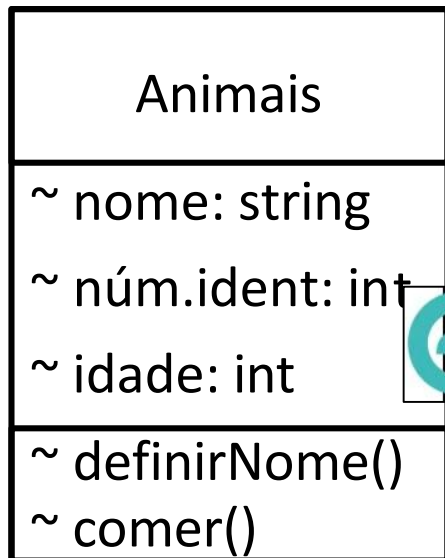


# Sistema Zoológico - Visibilidade

Pode ser **usado por qualquer outra classe**, contanto que esteja no mesmo pacote.

## Pacote

**É raramente usado**

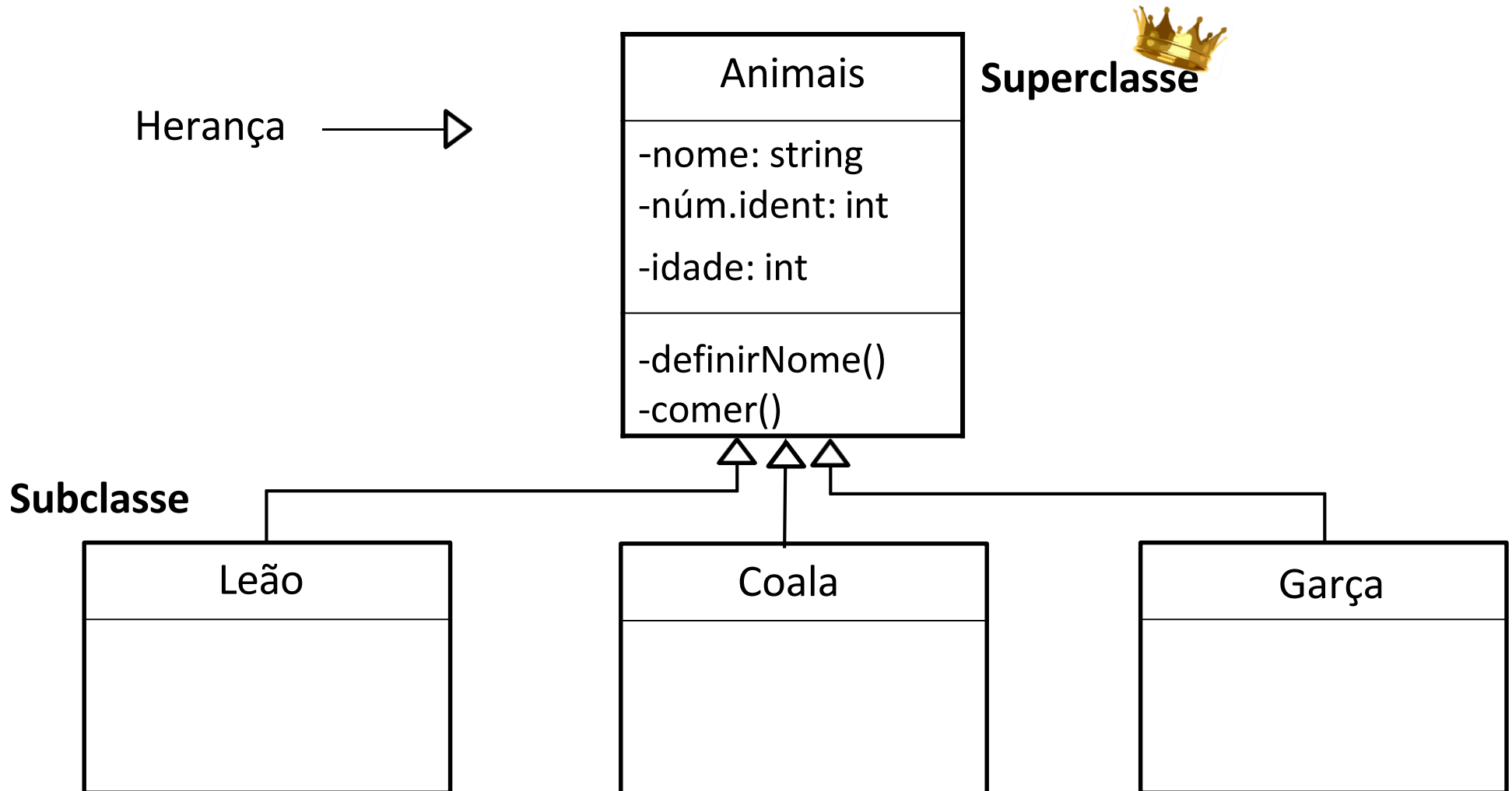


## Visibilidade

- privado (atributos)
- + público (métodos)
- # protegido (atributos)
- ~ pacote/padrão

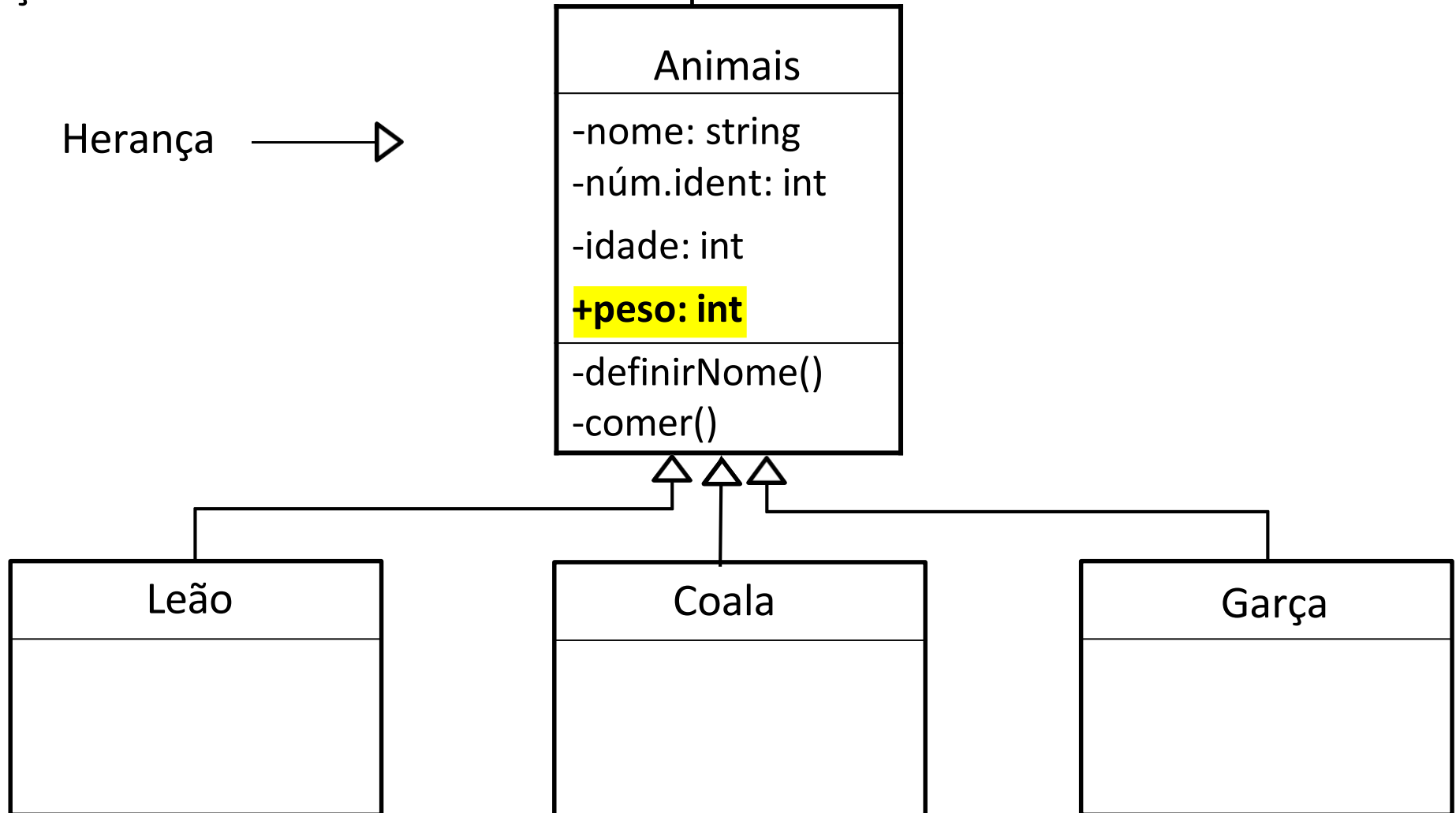
# Sistema Zoológico - Herança

Ao transformar a classe Animal em subclasses comunicamos que essas **subclasses herdam todos os atributos e métodos da classe Animal**.



# Sistema Zoológico - Herança

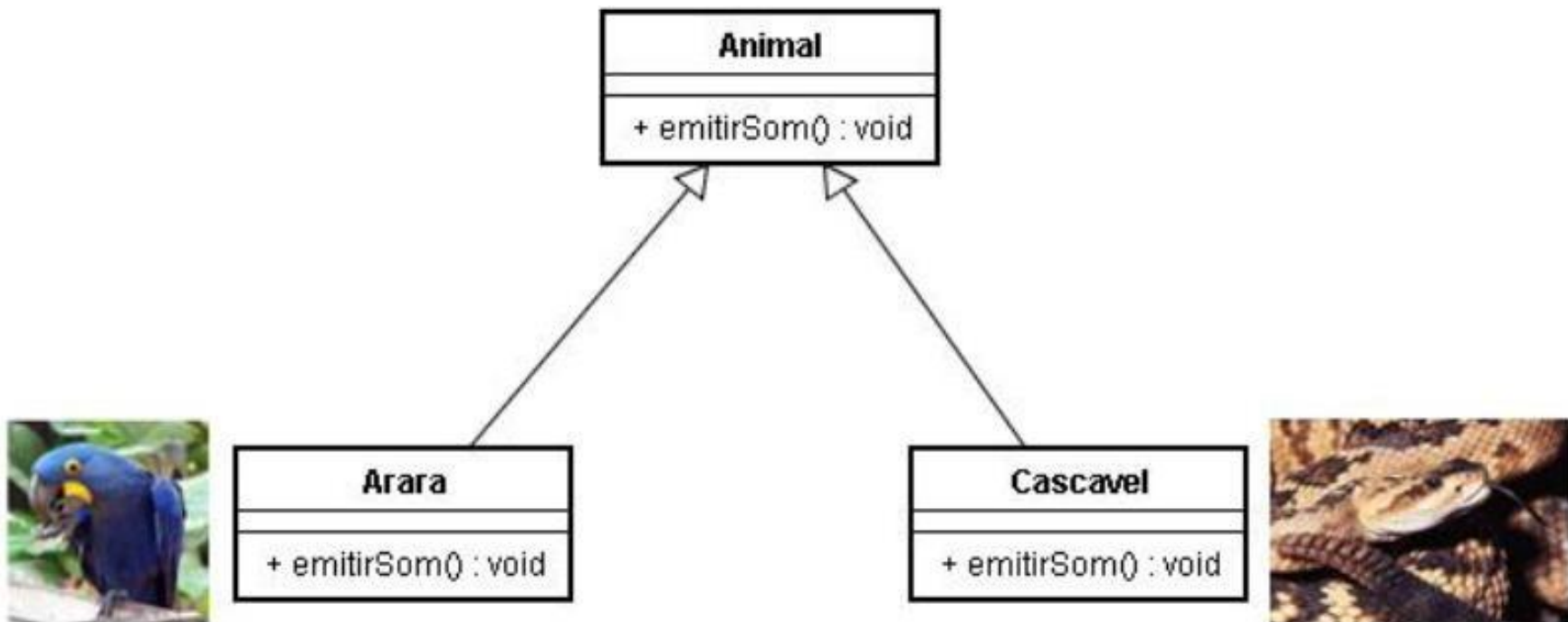
Se quisermos adicionar **o atributo peso** para todas as subclasse, basta fazer a mudança **somente na classe animal** e ela aplica-se a todas as outras classes.





# Polimorfismo

- Seu conceito está associado ao de Herança;
- Trabalha com a redeclaração de métodos previamente herdados por uma classe;
- Os métodos, **apesar de semelhantes, diferem de alguma forma** da implementação utilizada na superclasse. Assim, é necessário a implementação na subclasse:



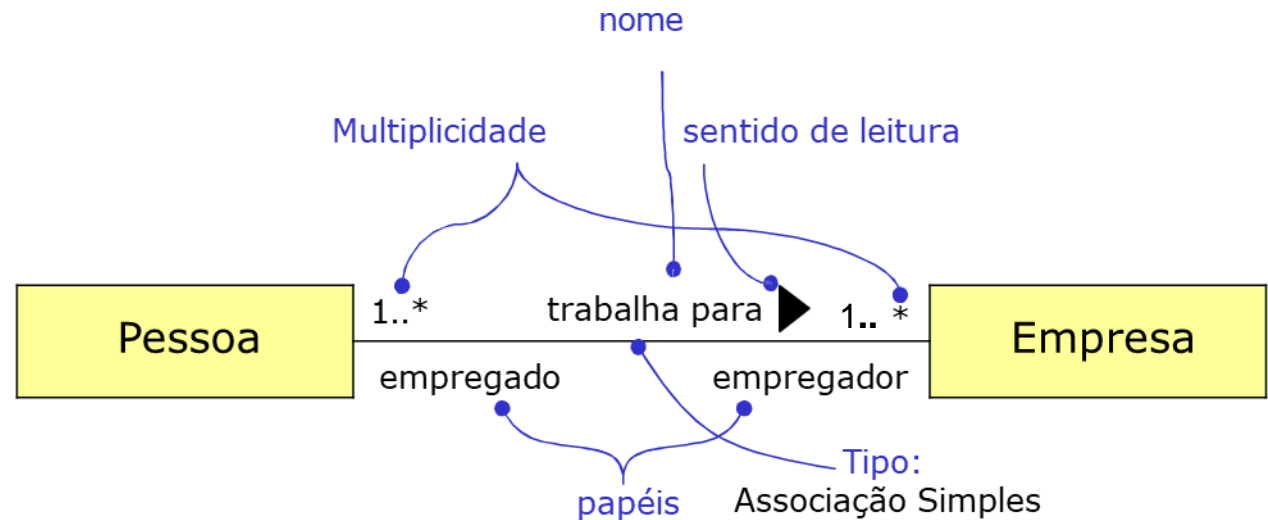
# Encapsulamento

- Em Programação Orientada a Objetos, significa **separar o programa em partes, o mais isoladas possível**;
- O encapsulamento almeja tornar o software mais flexível, fácil de alterar e de criar novas implementações;
- Quando houver código duplicado é recomendado procurar um lugar para encapsulá-lo.



# Multiplicidade

- Uma pessoa pode trabalhar para uma ou várias empresas (1..N).
- Uma empresa pode empregar várias pessoas (1..N).



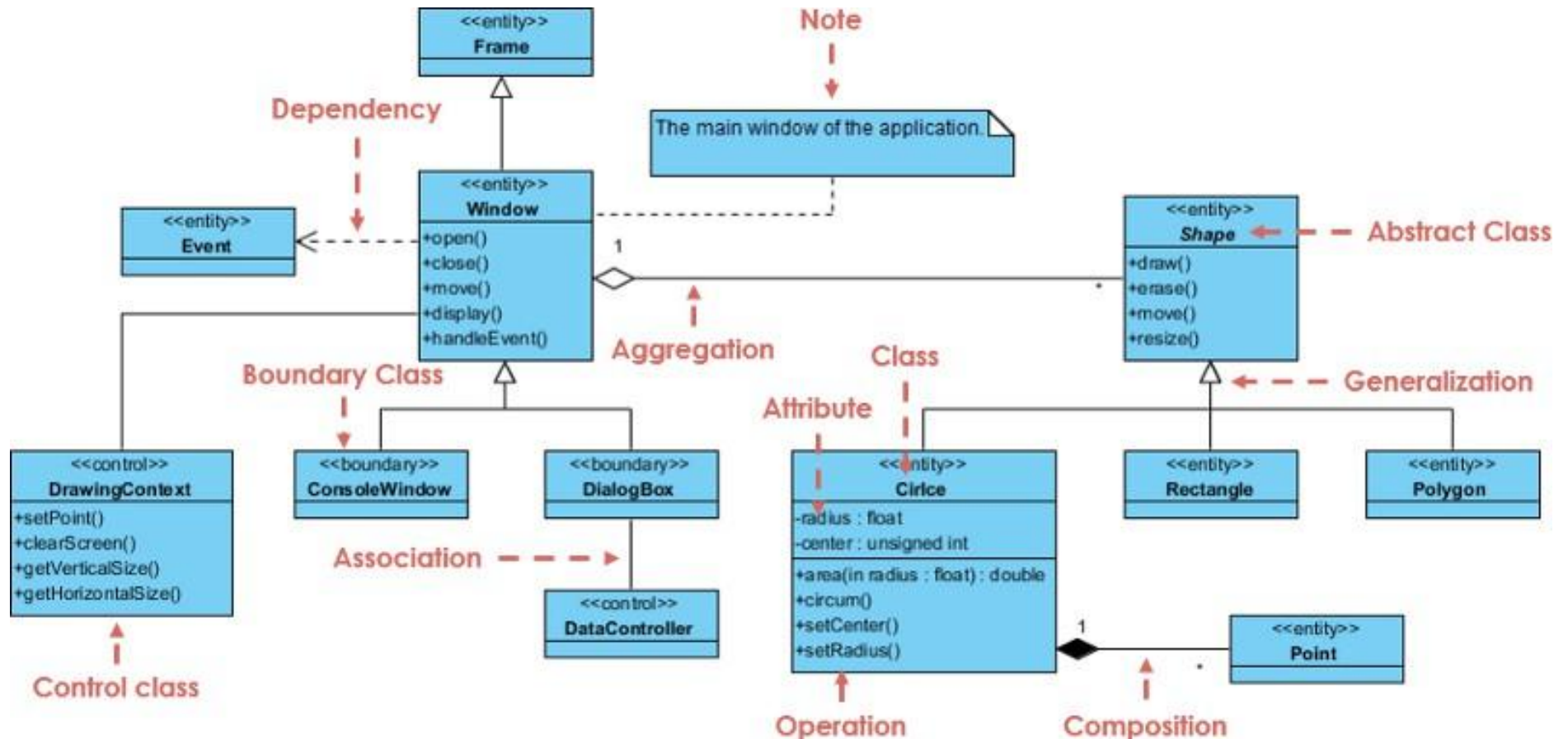
## Indicadores de multiplicidades:

- 1 Exatamente um
- 1..\* Um ou mais
- 0..\* Zero ou mais (muitos)
- \* Zero ou mais (muitos)
- 0..1 Zero ou um
- m..n Faixa de valores (por exemplo: 4..7)

**E a navegabilidade?**

# Diagrama de Classe

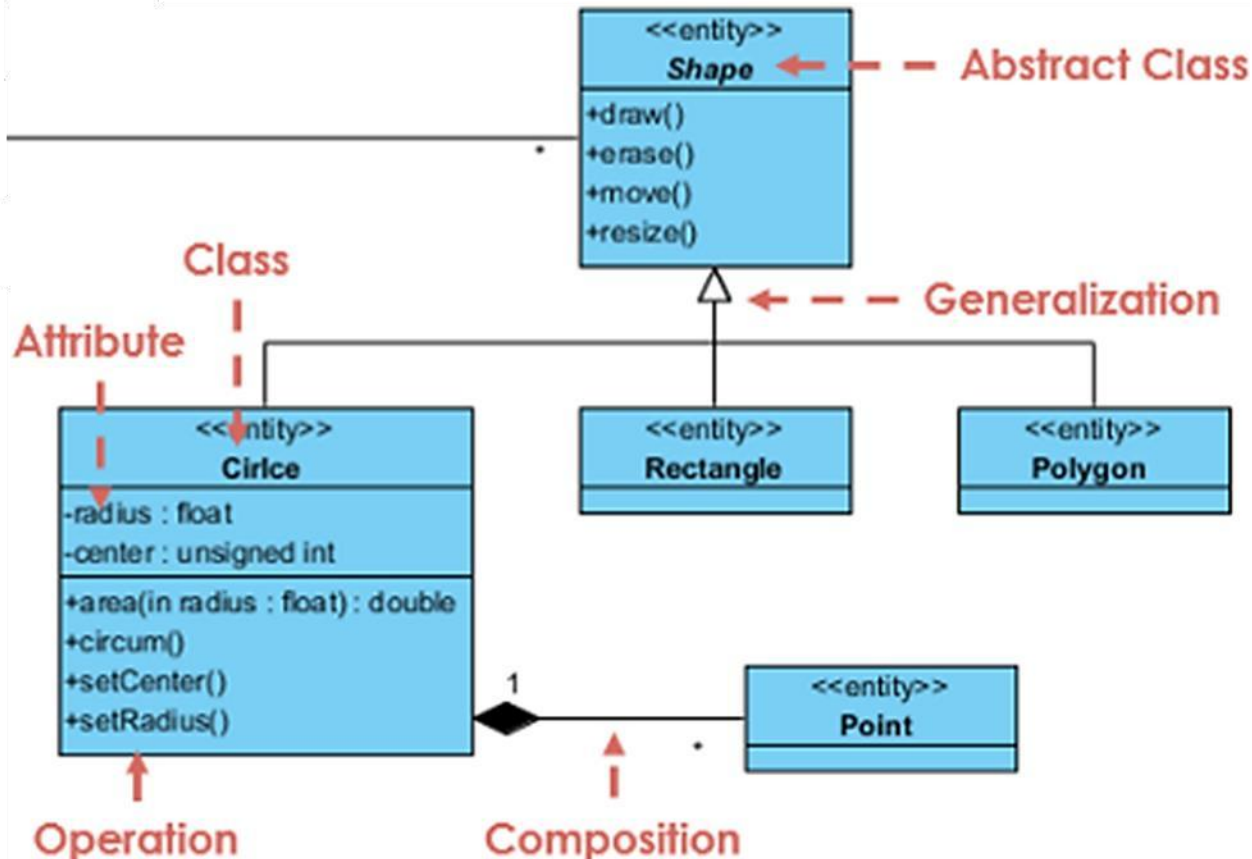
Os diagramas de classes podem ser acompanhados por **anotações de classes ou relacionamentos**.



# Diagrama de Classe

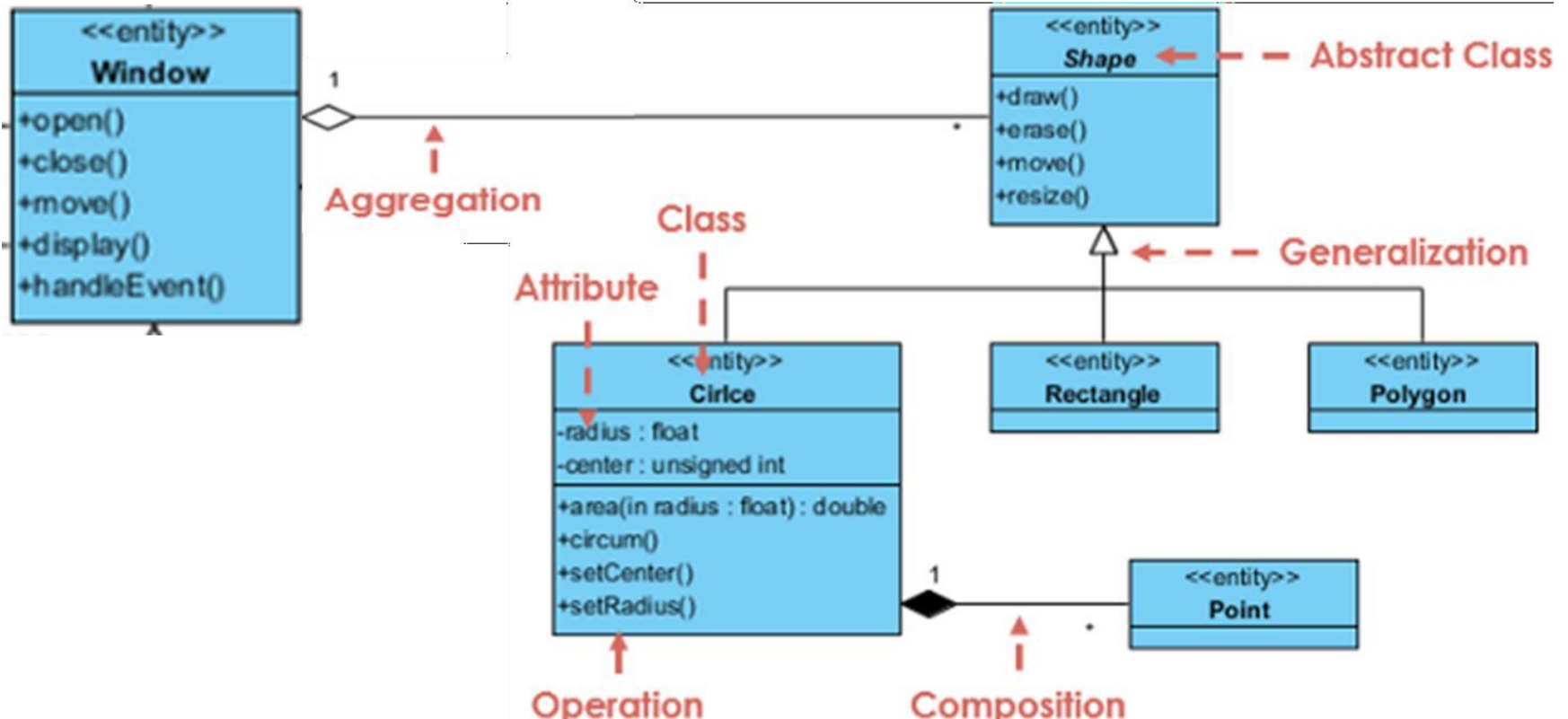
- Shape é uma classe abstrata. É mostrado em itálico.
- Shape é uma **superclasse**.
- Círculo, Retângulo e Polígono são derivados de Formas.

Podemos ver uma relação de generalização/herança.



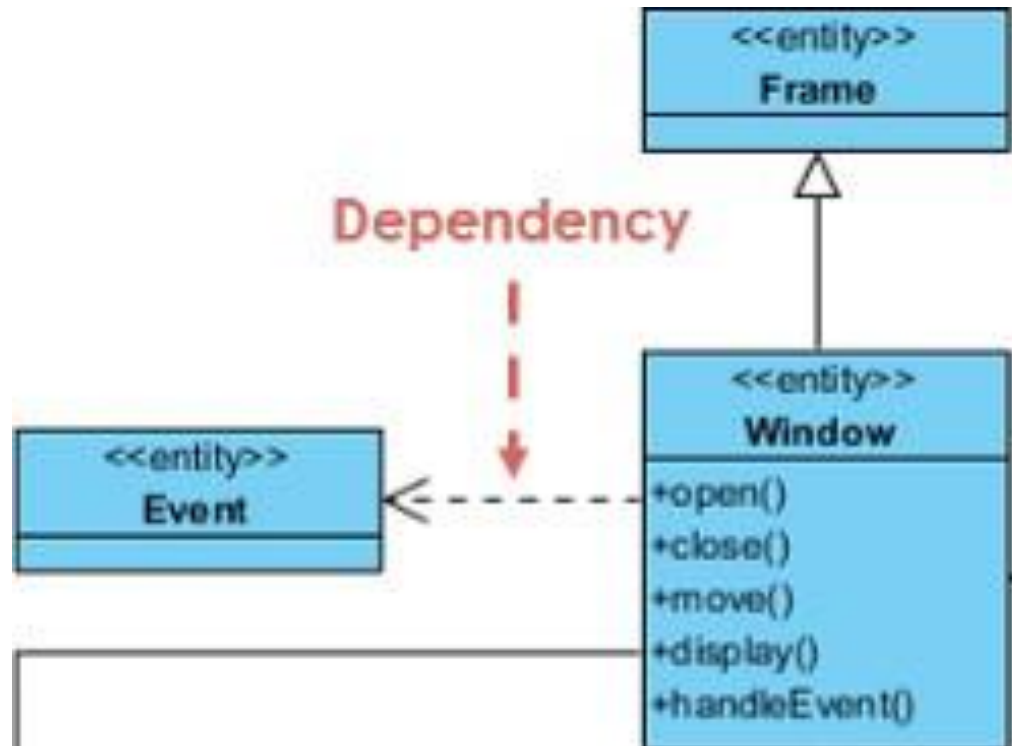
# Diagrama de Classe

- O Shape faz parte da Windows. Este é um relacionamento de **Agregação simples**. As formas (Shape) podem existir sem janelas (Windows).
- Um Point faz parte de um círculo. Este é um **Agregação de composição**. Sem círculos, os pontos não podem existir.



# Diagrama de Classe

- O Windows depende de Event. No entanto, Event não depende de Window.



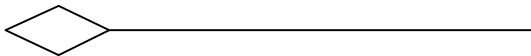
# Relacionamento - Resumo

- Elementos de um diagrama de classes - **Relacionamento**

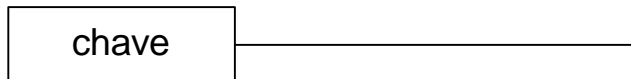
Associação Bidirecional



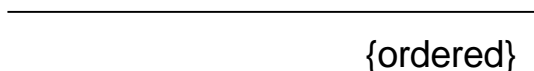
Agregação



Associação Qualificada



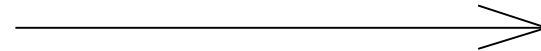
Associação Ordenada



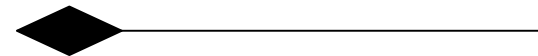
Realização



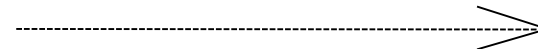
Associação Unidirecional



Composição



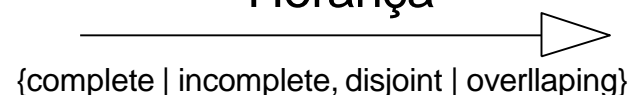
Dependência



Associação Exclusiva



Herança





# Diagrama de Sequência

## Definição

Diagramas de Seqüência são utilizados para representar e modelar o fluxo de mensagens, eventos e ações entre objetos e componentes de um sistema.

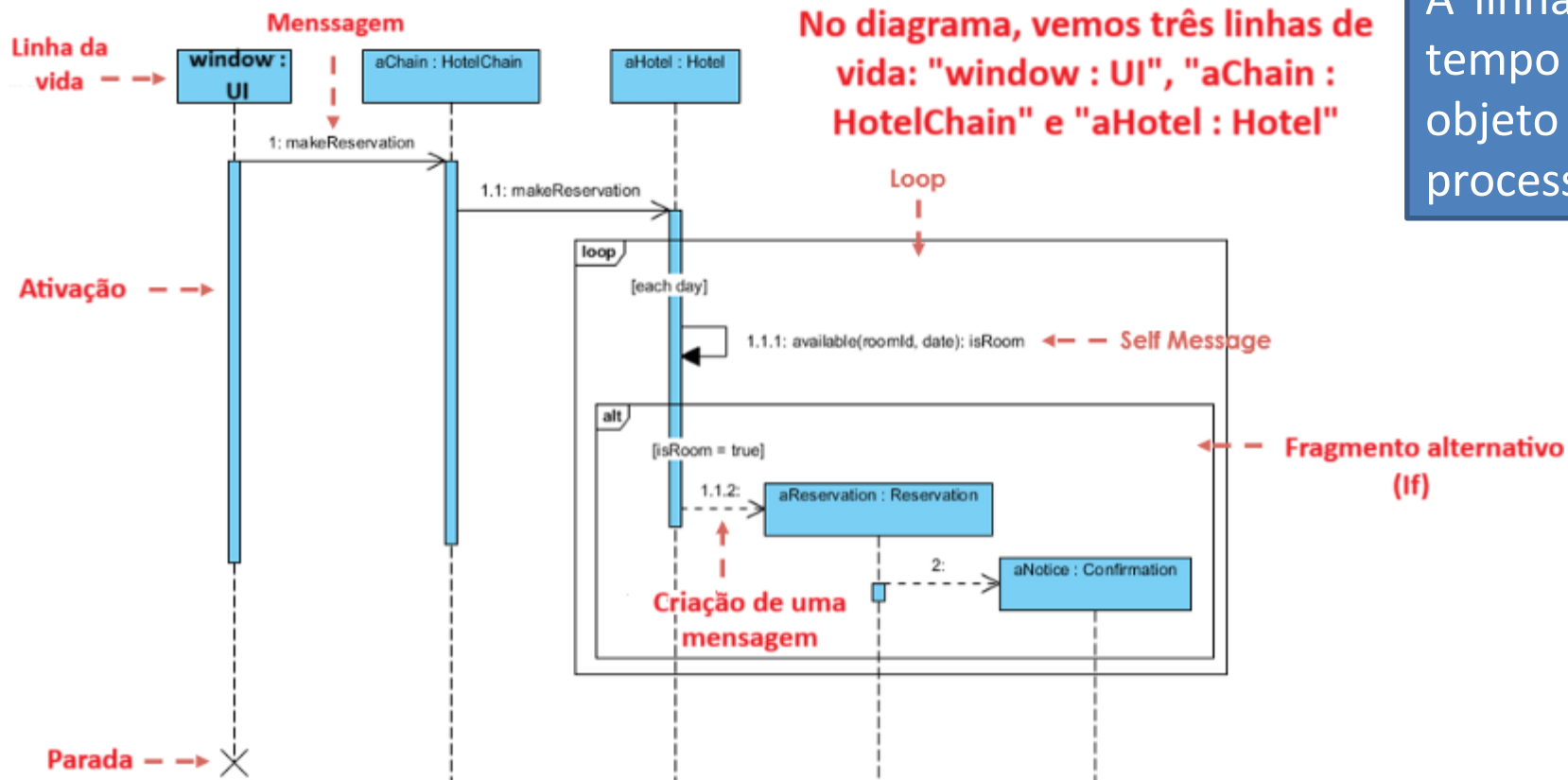
- Descreve a seqüência de mensagens que devem ser trocadas para realizar um determinado cenário.
- São também utilizados para modelar a arquitetura dos sistemas

# Diagrama de Sequência

- São diagramas de interação que detalham como uma ação é executada. Eles capturam interações entre objetos em um contexto de colaboração.
- Os diagramas de sequência são focados no **tempo** e mostram visualmente a **sequência de interações** usando o **eixo vertical** do diagrama para representar o tempo, quais **mensagens** são enviadas e quando.
- O tempo aumenta conforme você desce na página. Os objetos envolvidos na operação são listados da **esquerda para a direita** de acordo com o tempo em que participaram da sequência de **mensagens**.

# Diagrama de Sequência – Linha da Vida

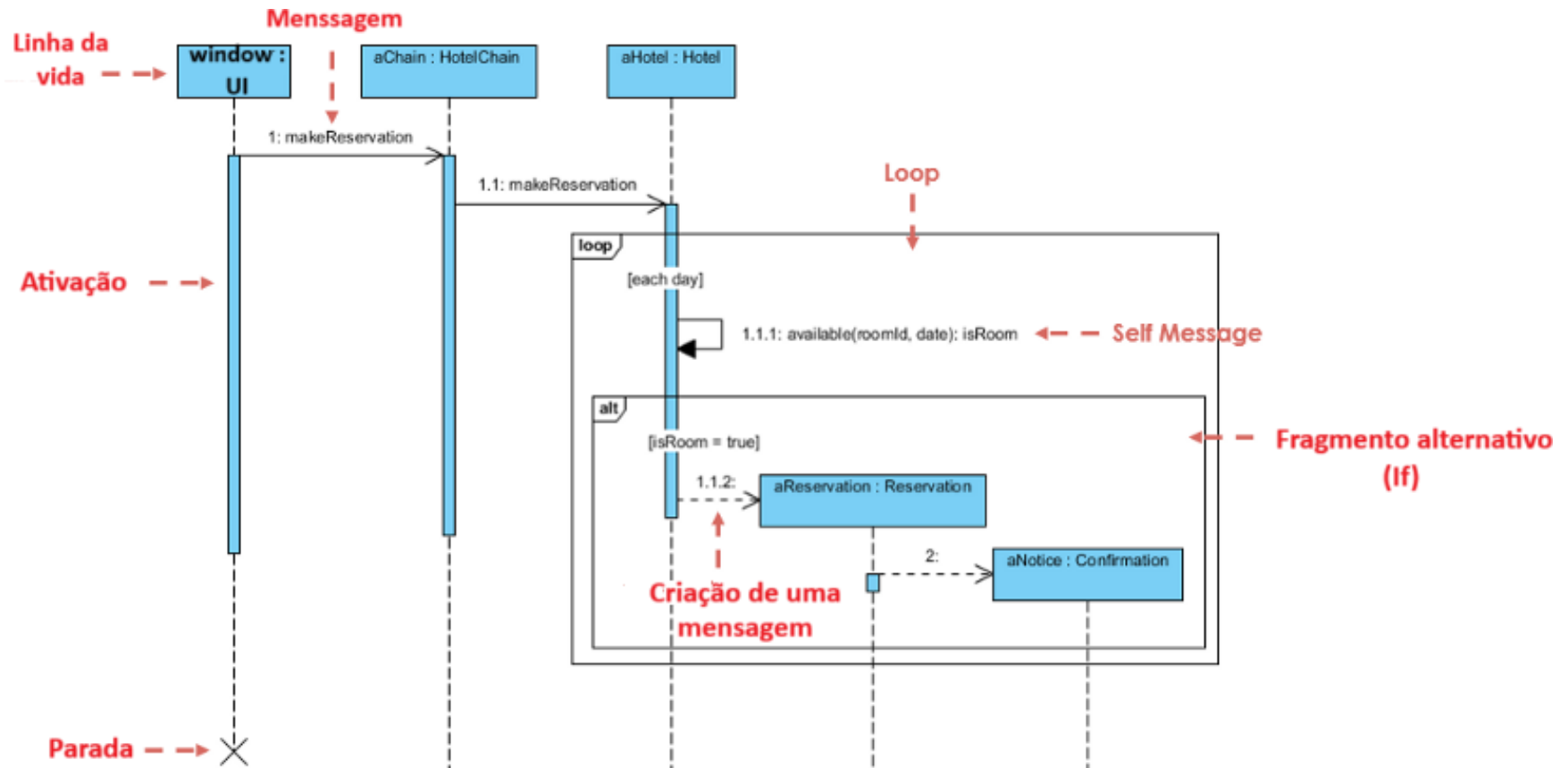
Este diagrama de sequência representa o processo de fazer uma reserva em um sistema de reservas de hotel, mostrando a interação entre diferentes componentes e objetos.



A linha de vida marca o tempo durante o qual o objeto está ativo no processo.

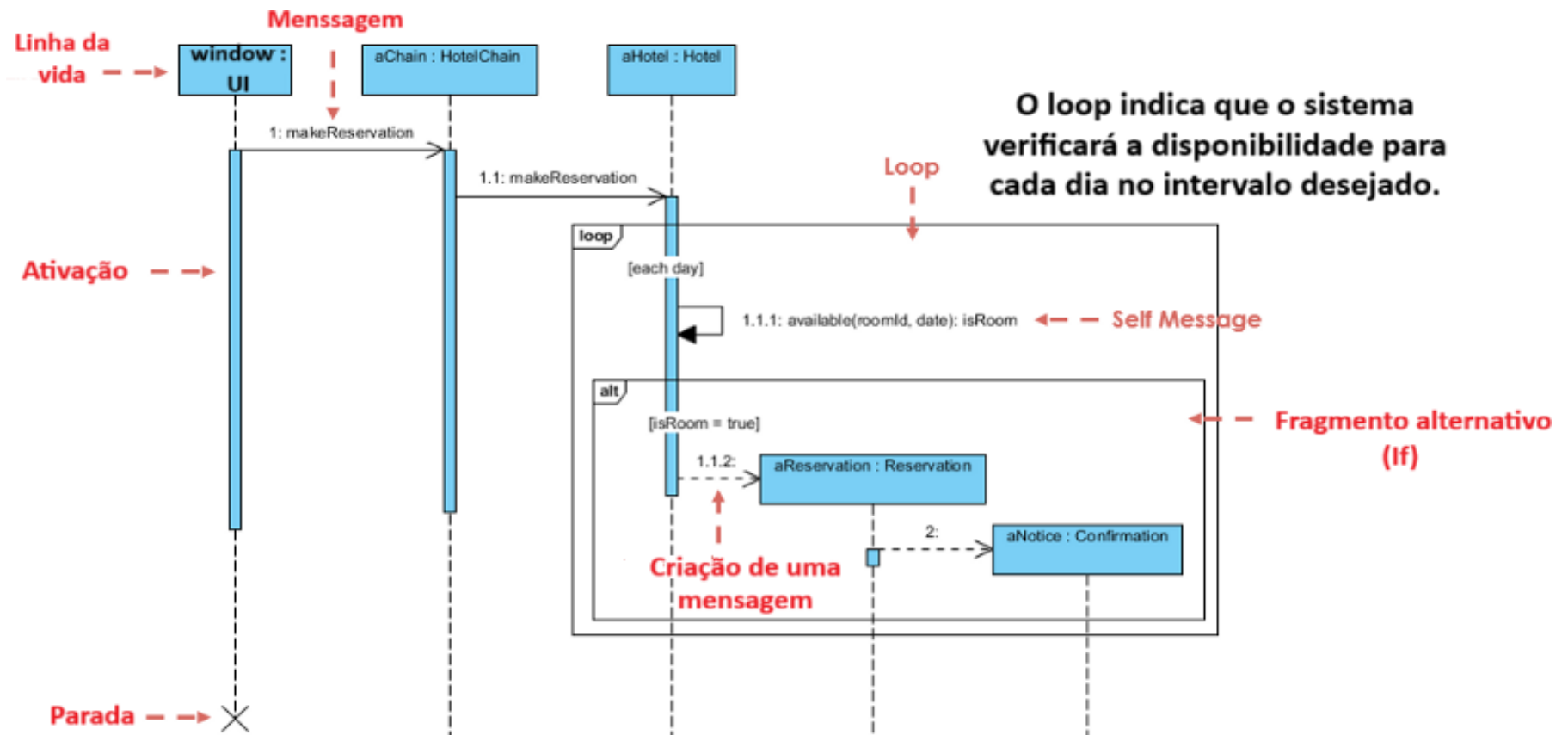
# Diagrama de Sequência – 1º Interação

A primeira interação é quando a interface do usuário ("window : UI") envia uma mensagem de "makeReservation" para a cadeia de hotéis ("aChain : HotelChain"), indicando que o usuário está tentando fazer uma reserva.



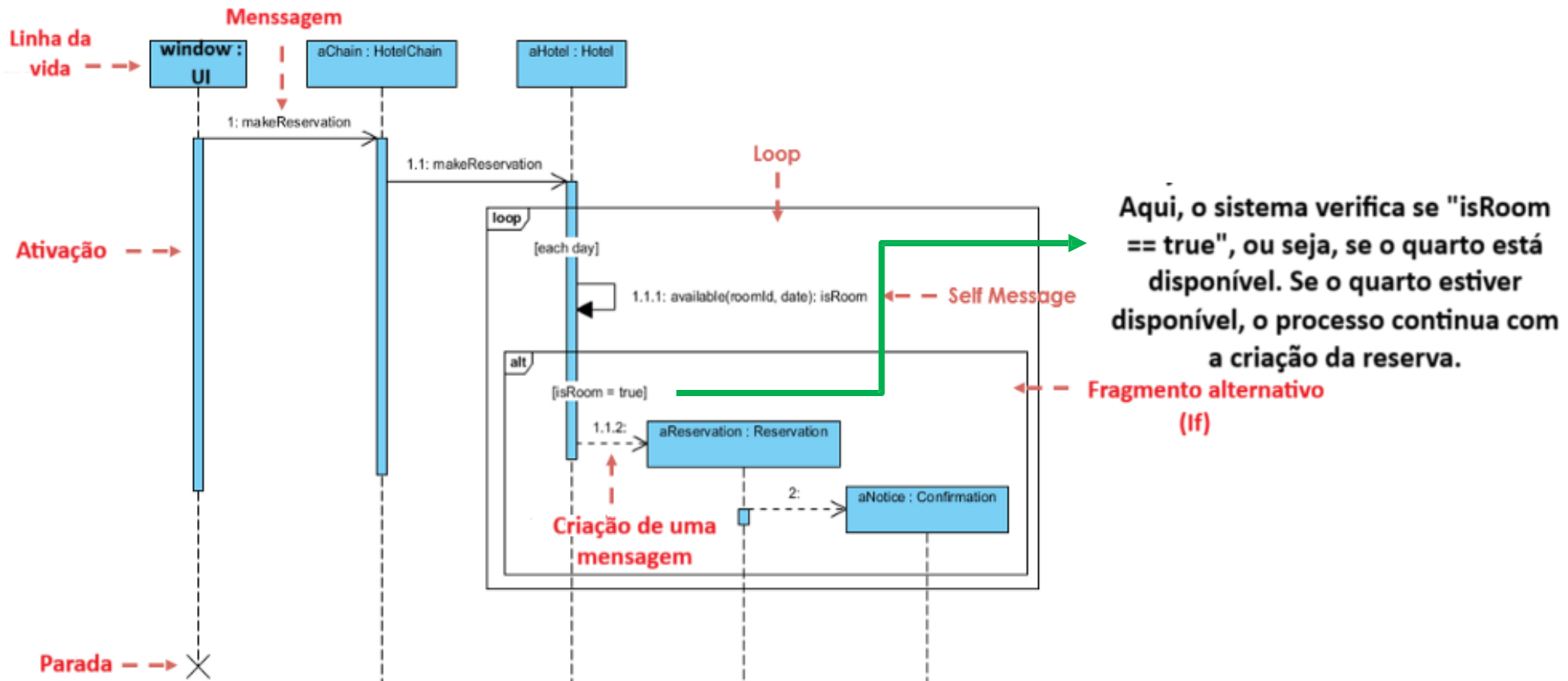
# Diagrama de Sequência – Loop

No diagrama de sequência há um fragmento combinado de loop. Podemos ver dentro do loop a mensagem "available(roomId, date)" é enviada para verificar se o quarto está disponível em um determinado dia.



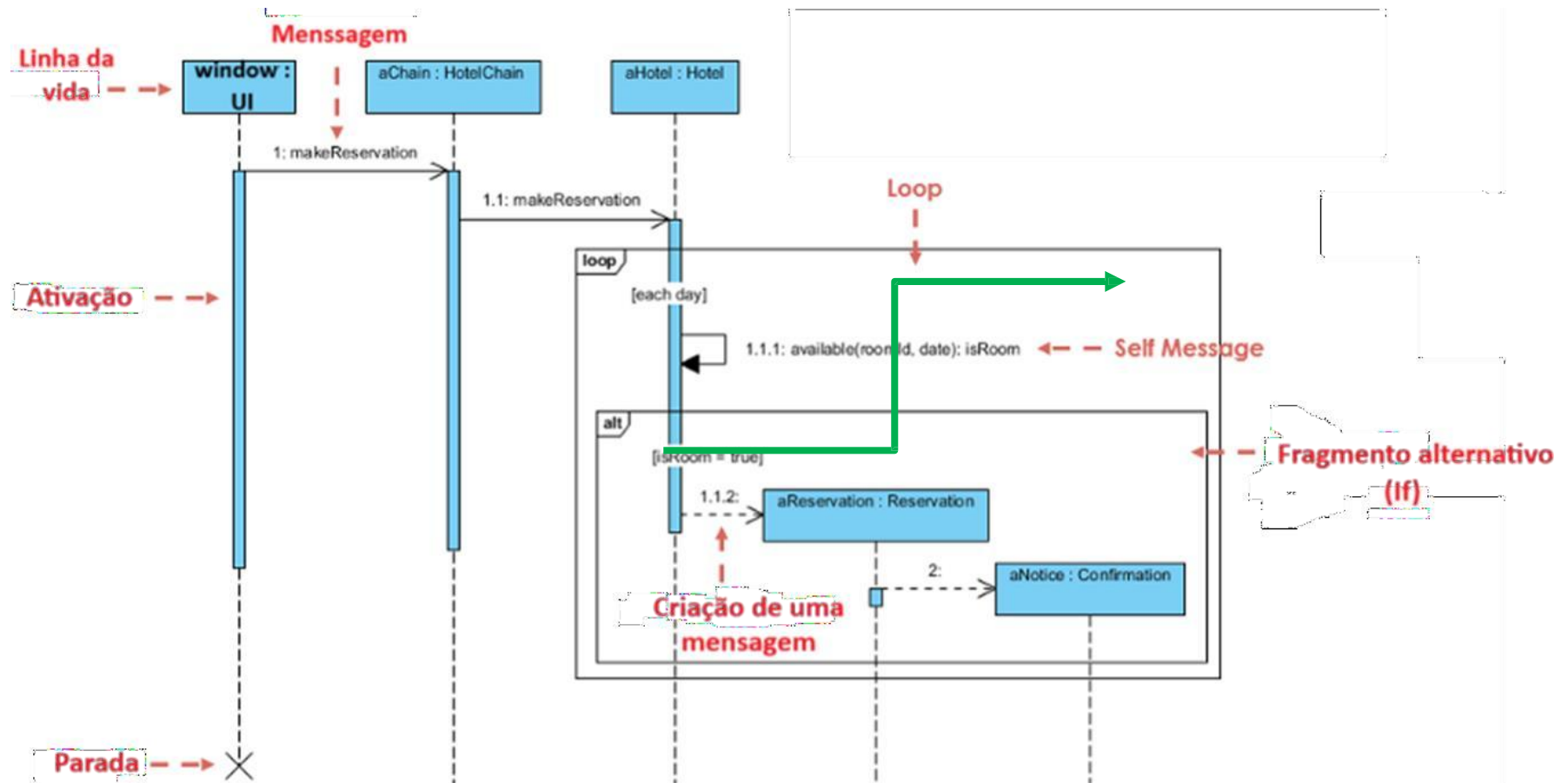
# Diagrama de Sequência – Fragmento Alternativo

O fragmento alternativo indica uma bifurcação no fluxo, ou seja, um “se” condicional.



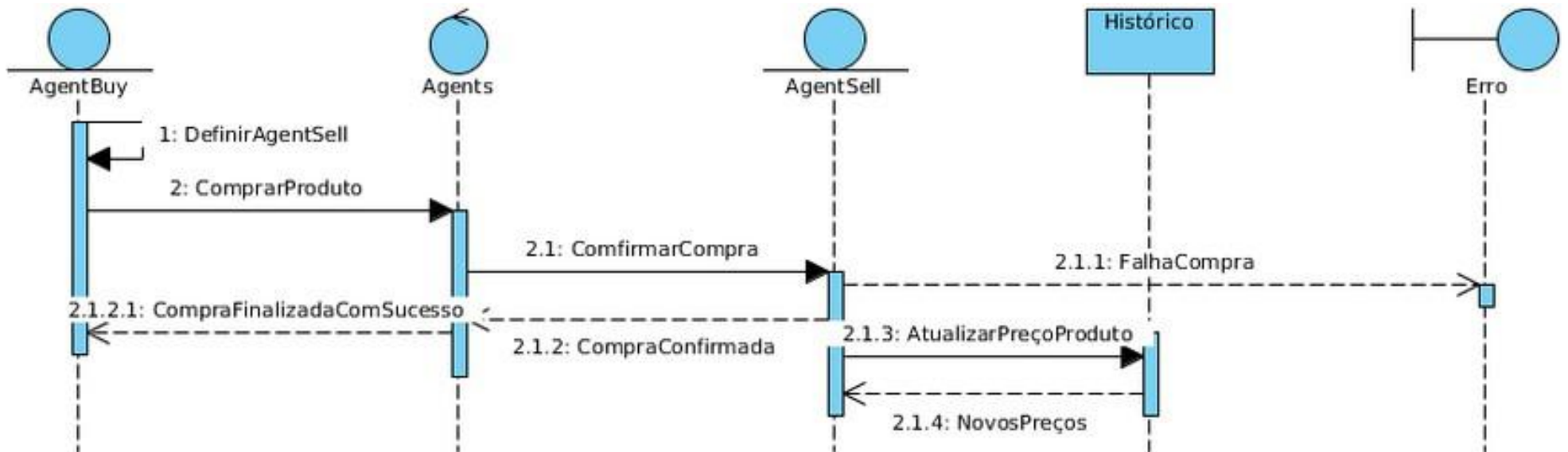
# Diagrama de Sequência – Mensagem

Uma vez que o quarto está disponível, é criada uma reserva ("aReservation : Reservation") e uma confirmação é enviada ("aNotice : Confirmation")



# Descrição do Problema - Efetivando a Venda

Este diagrama de sequência representa o agente de compra efetivando a venda.



- **AgentBuy** : O agente que inicia uma compra.
- **Agentes** : Representa um conjunto de agentes ou um intermediário.
- **AgentSell** : O agente responsável pela venda do produto.
- **Histórico** : Um componente que armazena informações de falhas ou histórico de transações.
- **Erro** : Uma entidade ou componente que trata de erros ou falhas.



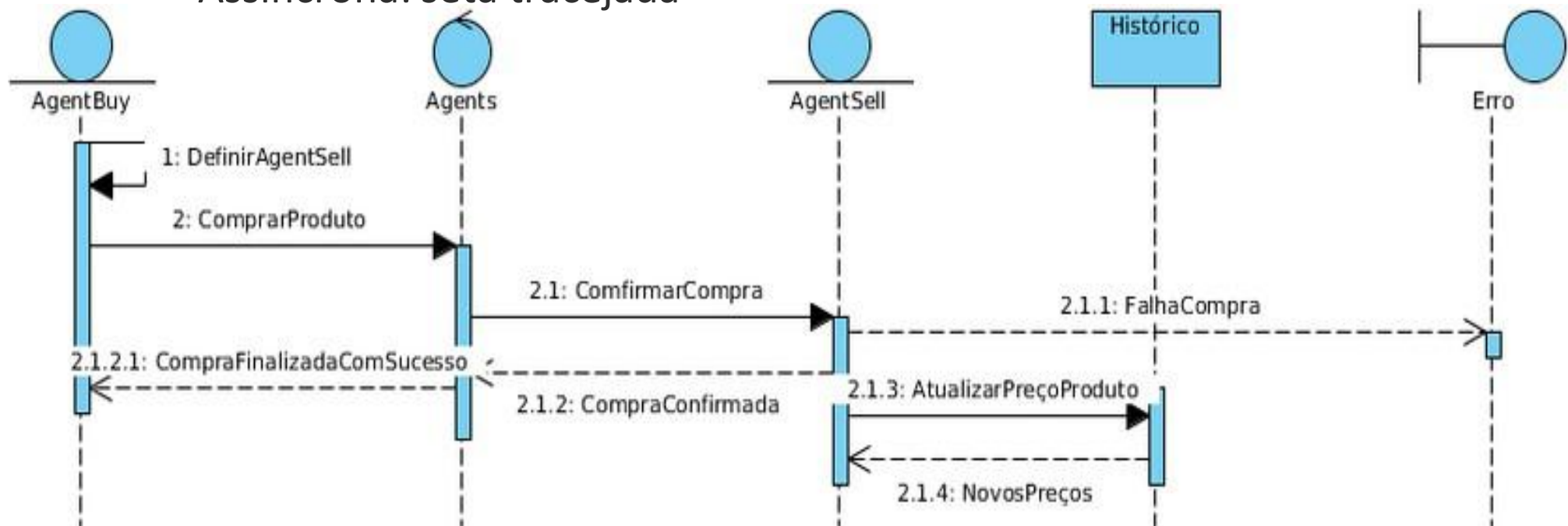
# Descrição do Problema - Efetivando a Venda

## Fluxo Geral:

O **AgentBuy** inicia o processo de definição do **AgentSell**, ele então solicita aos **Agentes** que comprem um produto.

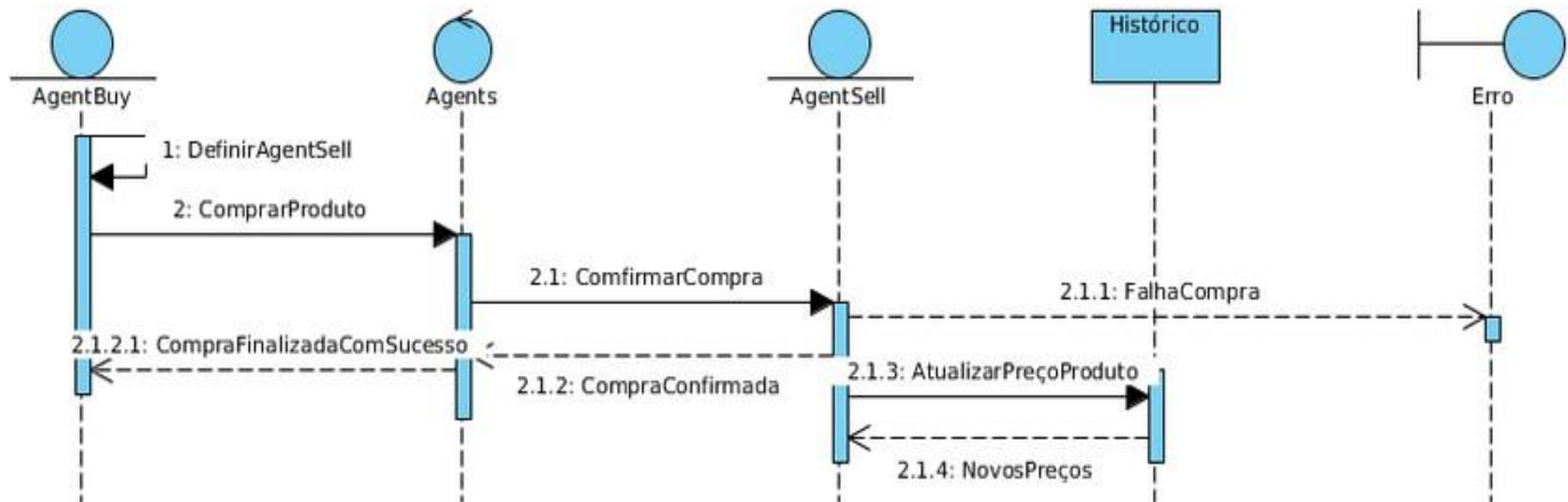
Tipo de mensagem:

- Síncrona: seta cheia
- Assíncrona: seta tracejada



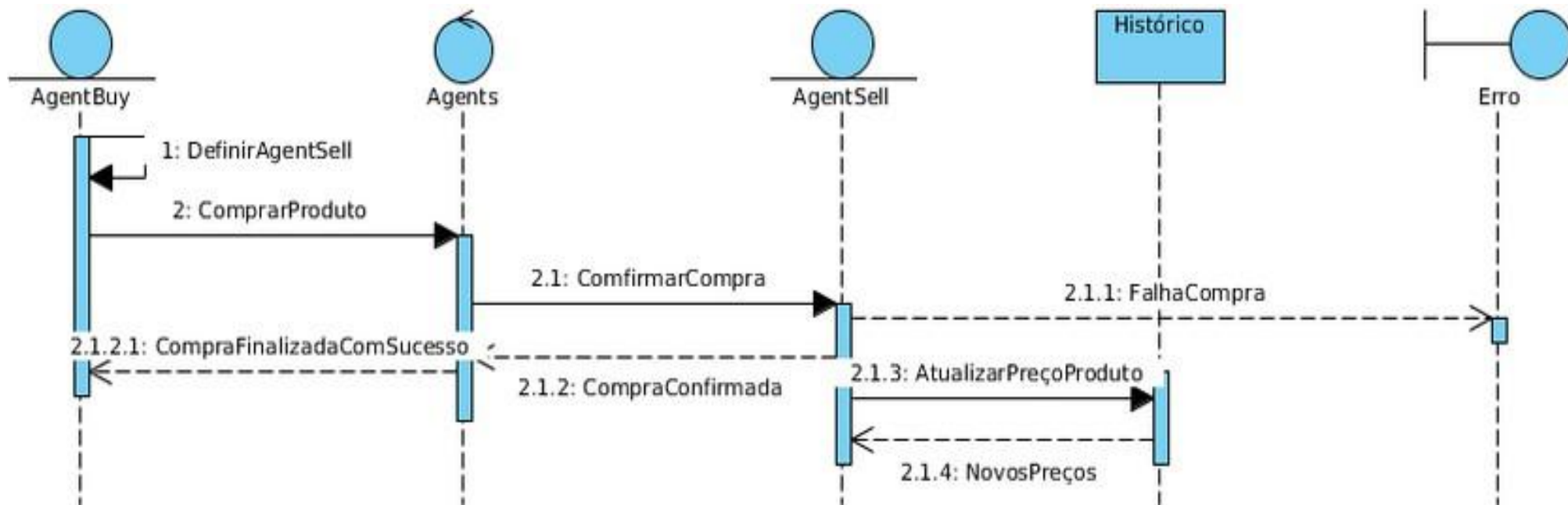
# Descrição do Problema - Efetivando a Venda

- O **Agents** tenta confirmar a compra com o **AgentSell** (2.1).
  - ✓ Se a compra falhar, uma mensagem de falha é registrada no **Histórico** e enviada à entidade de **Erro**.
  - ✓ Se a compra for bem-sucedida, a mensagem de sucesso é enviada de volta ao **AgentBuy**.



# Descrição do Problema - Efetivando a Venda

- Após uma compra bem-sucedida, há uma comunicação para que o AgentSell **atualize o preço do produto**.



# Diagrama Entidade Relacionamento

- É a quarta ferramenta importante para o projeto de software.
- O Diagrama Entidade-Relacionamento (DER) é uma ferramenta de modelagem útil para manipulação de banco de dados e arquivos.
- Seu principal propósito é representar os objetos de dados armazenados no software e suas respectivas relações.
- Ferramenta que ressalta os relacionamentos entre os depósitos de dados de um DFD.
- O DER é formado por vários elementos, tais como Entidades, Atributos e Relacionamentos.

# DER - Simbologia

## Entidade

- Uma entidade é um grupo de objetos que possuem as mesmas características.
- No DFD está associada a um depósito de dados.
- É representada por um retângulo, sendo as mesmas identificadas por um nome.

### **Exemplo** (Sistema Biblioteca):

**Usuários**

**Livros**

**Empréstimos**

## DER - Simbologia

### Entidade (Representação de Banco de Dados)

#### — Entidade Usuários (Sistema Biblioteca)

- **USUÁRIOS** = #Nro\_USCS + Nome + Tipo + Status + Qdade

#Nro_USCS	Nome	Tipo	Status	Qdade
343254	MARCELO LINCE	PROFESSOR	OK	7
453453	JOÃO DA SILVA	FUNCIONÁRIO	OK	2
568654	MARIA LUCIA GOMES	FUNCIONÁRIO	NÃO OK	4
654534	JOSÉ PAULINO MAIA	ALUNO	OK	6
(...)	(...)	(...)	(...)	(...)

#### — Entidade Livros (Sistema Biblioteca)

- **LIVROS** = #Código + Título + Disponibilidade + (Local)

#Código	Título	Disponibilidade	Local
34234545	FÍSICA GERAL II	SIM	PRAT05-A
66545656	NEURAL NETWORKS	NÃO	PRAT06-B
98343442	FUZZY SETS	SIM	PRAT03-A
(...)	(...)	(...)	(...)

# DER - Entidade (Atributos, Registros e Dados)

## – Atributos

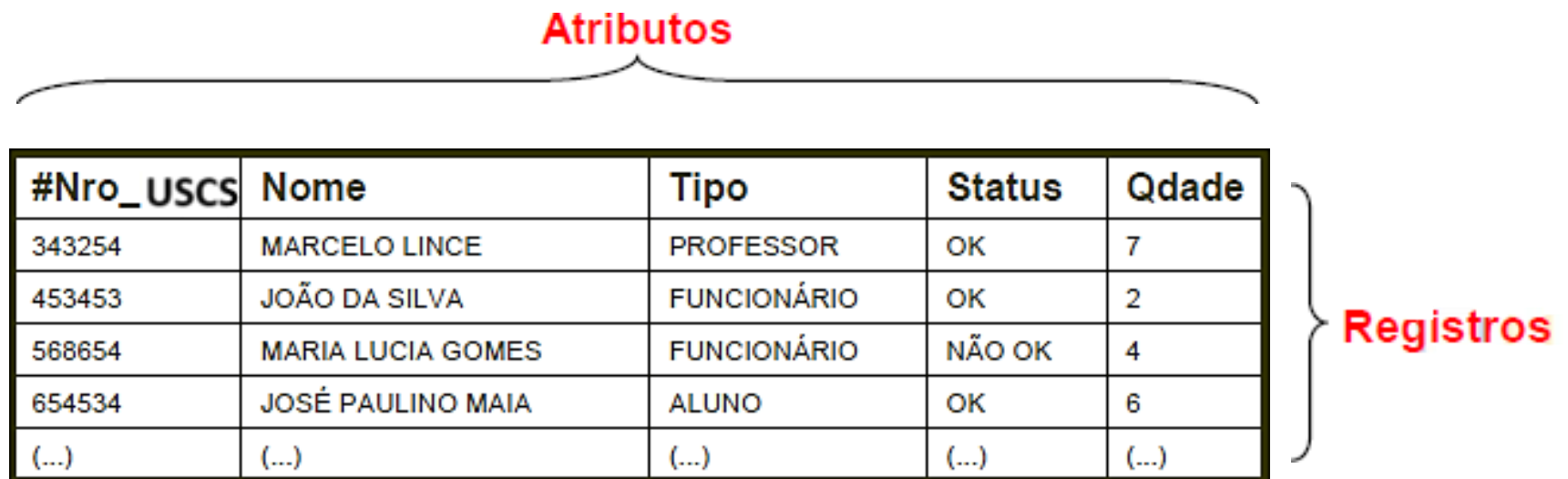
- Definem os diversos campos que contem a tabela de Entidade. São as colunas da tabela.

## – Registros

- São as diversas coleções que são formadas individualmente por todos os atributos. São as linhas da tabela.

## – Dados

- São os valores inseridos em cada atributo da tabela.



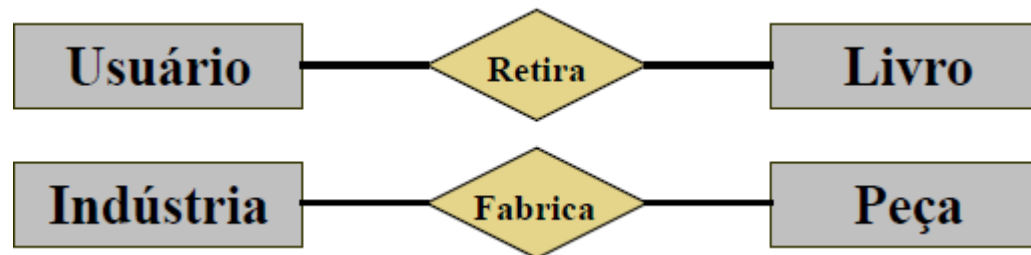
The diagram illustrates the structure of a data table. A horizontal curly brace above the table columns is labeled "Atributos" in red. A vertical curly brace to the right of the table rows is labeled "Registros" in red.

#Nro_USCS	Nome	Tipo	Status	Qdade
343254	MARCELO LINCE	PROFESSOR	OK	7
453453	JOÃO DA SILVA	FUNCIONÁRIO	OK	2
568654	MARIA LUCIA GOMES	FUNCIONÁRIO	NÃO OK	4
654534	JOSÉ PAULINO MAIA	ALUNO	OK	6
(...)	(...)	(...)	(...)	(...)

# Relacionamentos no DER

- Relacionamento
  - Representa o conjunto de conexões existentes entre os diversos objetos.
    - São ligações somente entre as entidades que possuem um vínculo.
    - Cada instância do relacionamento representa uma associação entre zero ou mais ocorrências de uma entidade.
    - É representada por um losango e identificada por um verbo.

## Exemplos:





# Cardinalidade

- **Cardinalidade**

- É o número de ocorrências de um objeto relativo a uma entidade em referência a um objeto da entidade do lado oposto.
- É a ESSÊNCIA do Diagrama Entidade-Relacionamento.

- **Cardinalidade Mínima**

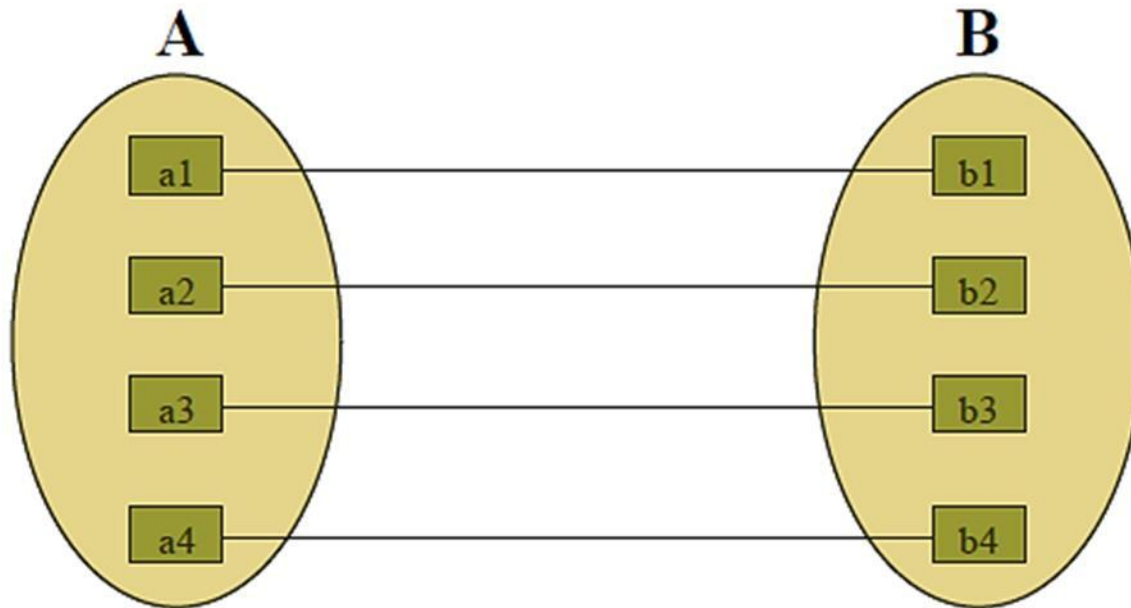
- Indica quantas ocorrências de uma Entidade participam no **MÍNIMO** do relacionamento com a outra entidade.
- Pode ter valor 0 ou 1.

- **Cardinalidade Máxima**

- Indica quantas ocorrências de uma Entidade participam no **MÁXIMO** do relacionamento com uma outra entidade.
- Pode ter valor 1 ou N.

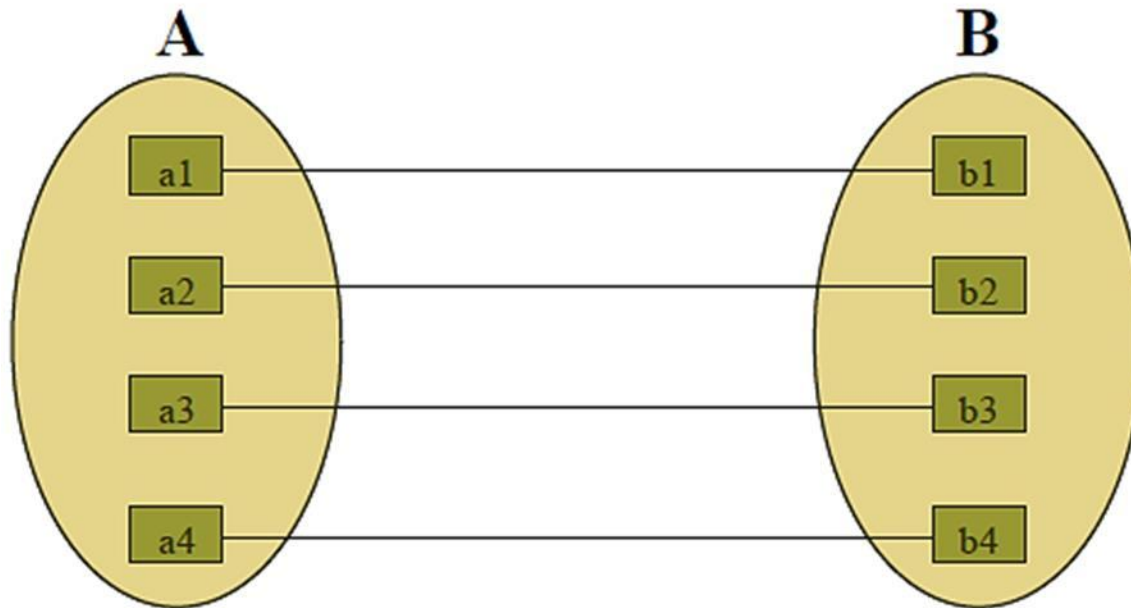
# Relacionamentos no DER

- **Relacionamentos “Um para Um” -> 1:1**
  - Cada entidade da classe “A” está associada a **UMA ÚNICA** entidade da classe “B”.
    - Cardinalidade Mínima -> 1
    - Cardinalidade Máxima -> 1



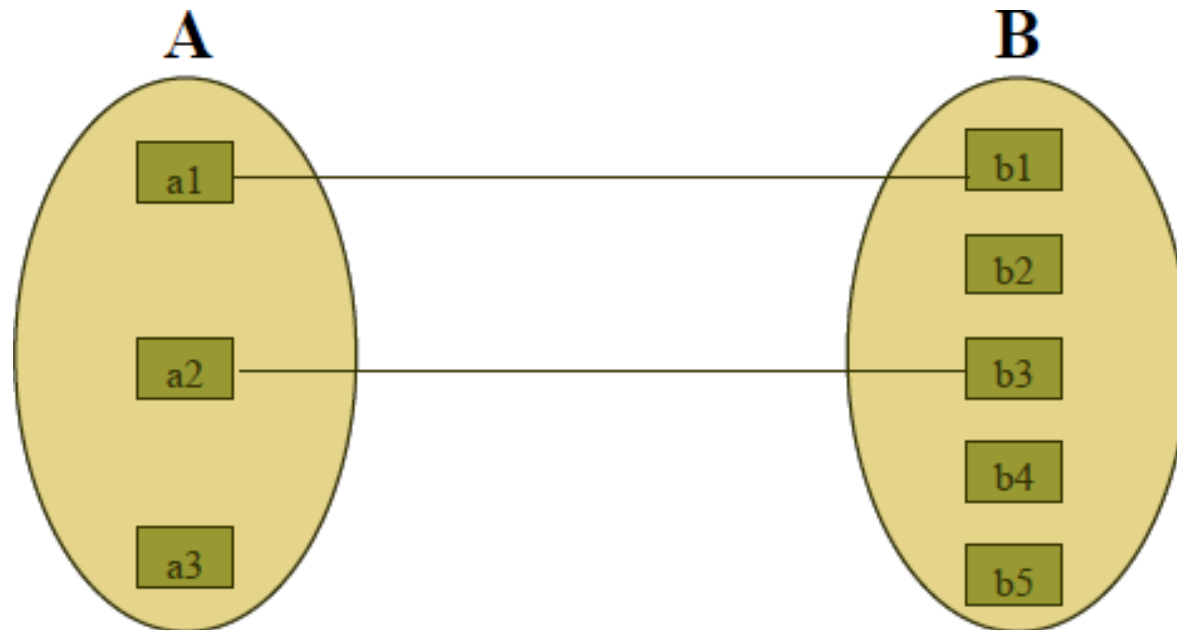
# Relacionamentos no DER

- **Relacionamentos “Um para Um” -> 1:1**
  - Cada entidade da classe “A” está associada a **UMA ÚNICA** entidade da classe “B”.
    - Cardinalidade Mínima -> 1
    - Cardinalidade Máxima -> 1



# Relacionamentos no DER

- **Relacionamentos “Zero para 1” -> 0:1**
  - Cada entidade da classe “A” está associada a **ZERO** ou a **UMA ÚNICA** ocorrência da classe “B”.
    - Cardinalidade Mínima -> 0
    - Cardinalidade Máxima -> 1

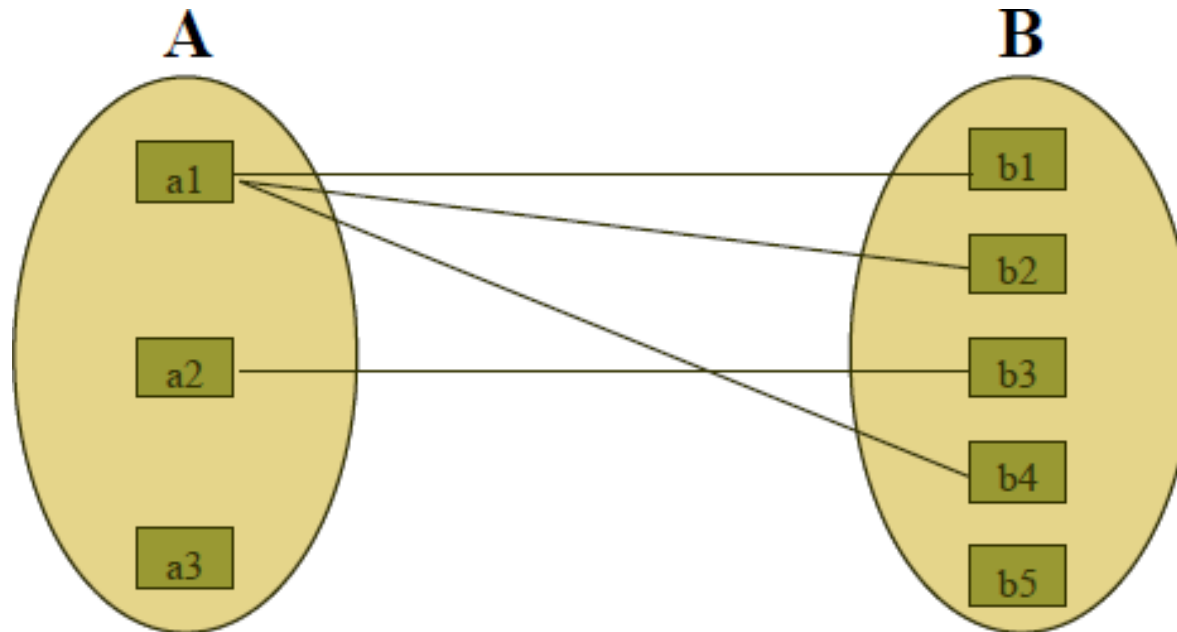


# Relacionamentos no DER

## Relacionamentos “Zero para Muitos” -> 0:N

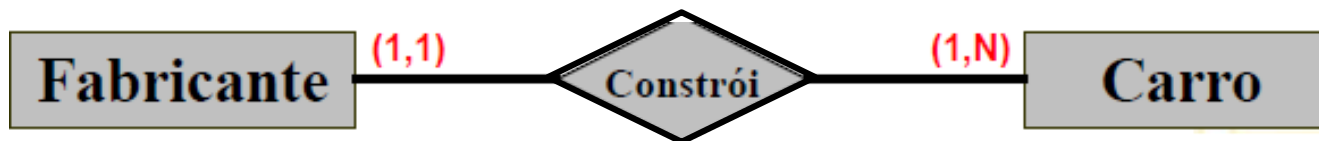
– Cada entidade da classe “A” está associada a **ZERO**, a **UMA** ou a **VÁRIAS** entidades da classe “B”.

- Cardinalidade Mínima -> 0
- Cardinalidade Máxima -> N



## Exemplo 1 de DER

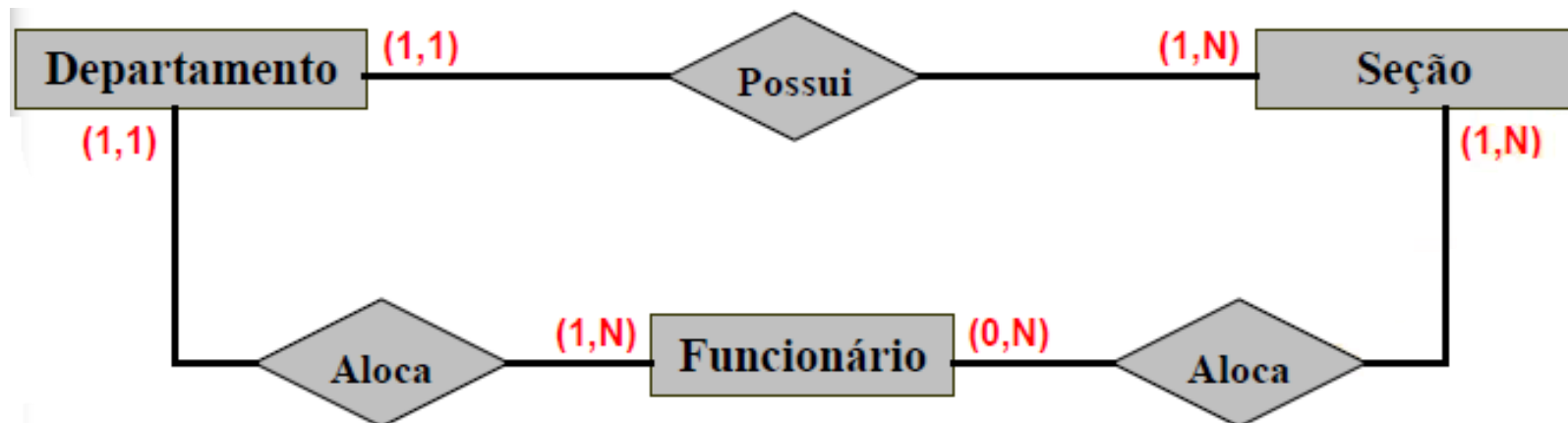
- **Seja duas entidades “Fabricante” e “Carro”, cujo relacionamento é o seguinte:**
  - Um carro só pode ser construído por um único fabricante.
  - Um fabricante pode construir mais de um carro.
- **Cardinalidades do Diagrama de Entidade- Relacionamento do Sistema:**
  - Fabricante constrói no mínimo 1 e no máximo N carros.
  - Carro pode ser construído por no mínimo 1 e no máximo 1 fabricante.



## Exemplo 2 de DER

Seja um banco de dados formado pelas entidades “Departamento”, “Seção” e “Funcionários”. As relações sobre este sistema são:

- Cada seção pertence a um único departamento.
- Cada departamento pode ser composto por no mínimo uma seção.
- Cada funcionário pode pertencer a uma ou mais seções.
- Cada seção pode ser composta por vários ou nenhum funcionários.
- Cada funcionário pode pertencer a apenas um departamento.
- Todo departamento tem no mínimo um funcionário.



# Chaves de Entidades - DER

É um conjunto de um ou mais atributos que, tomados coletivamente, permite-nos identificar unicamente um registro no conjunto-entidade.

## – Chaves Candidatas

- Conjuntos formados pelos **atributos** que permitem identificar unicamente um registro no conjunto-entidade.

## – Chave Primária

- Chave candidata **escolhida** como identificador de registros dentro do conjunto-entidade.
- Serve para evitar a redundância (repetição) dos dados e fazer o relacionamento entre os arquivos (tabelas).

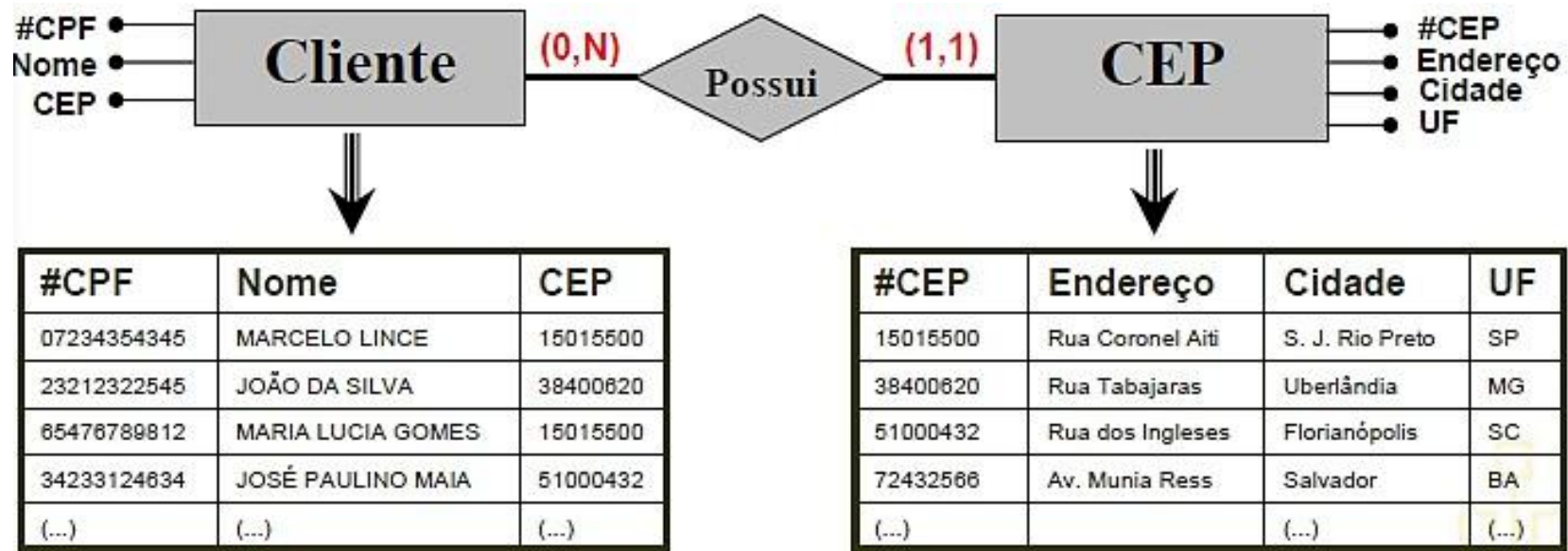
## – Chaves Estrangeira

- Atributo de uma entidade que é chave primária da outra entidade com a qual possui relacionamento.



# Chaves de Entidades - Exemplo

- Seja o seguinte relacionamento entre as entidades “Cliente” e “CEP”:



- ✓ **CPF:** Chave primária da Entidade-Cliente.
- ✓ **CEP:** Chave primária da Entidade-CEP e chave estrangeira da Entidade-Cliente.

# Referências

- [1]. SIERRA, Katy; BATES, Bert. Use a cabeça JAVA. Ed 2, Editora Altabooks.
- [2]. GUEDES, Gilleanes. UML Uma Abordagem Prática. Editora Novatec.
- [3]. FURLAN, José. Modelagem de Objetos através da UML. Editora Makron Books.
- [4]. CASTRO, Maurício. Orientação a Objetos. Solis/Univates (internet).
- [5]. BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. UML Guia do Usuário. Editora Campus.