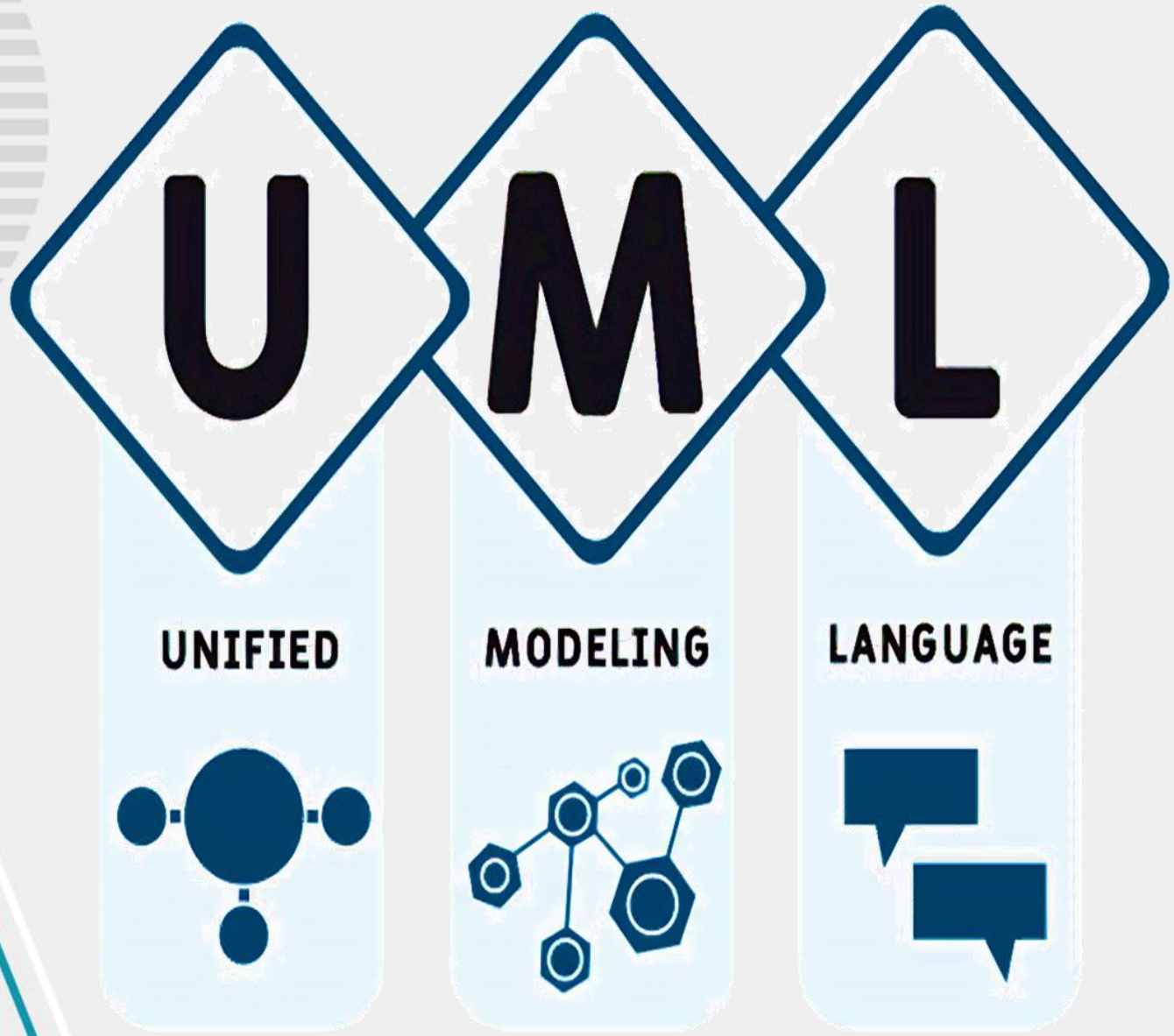
# BACK-END

## Desenvolvimento de Software para Internet

UNISUAM

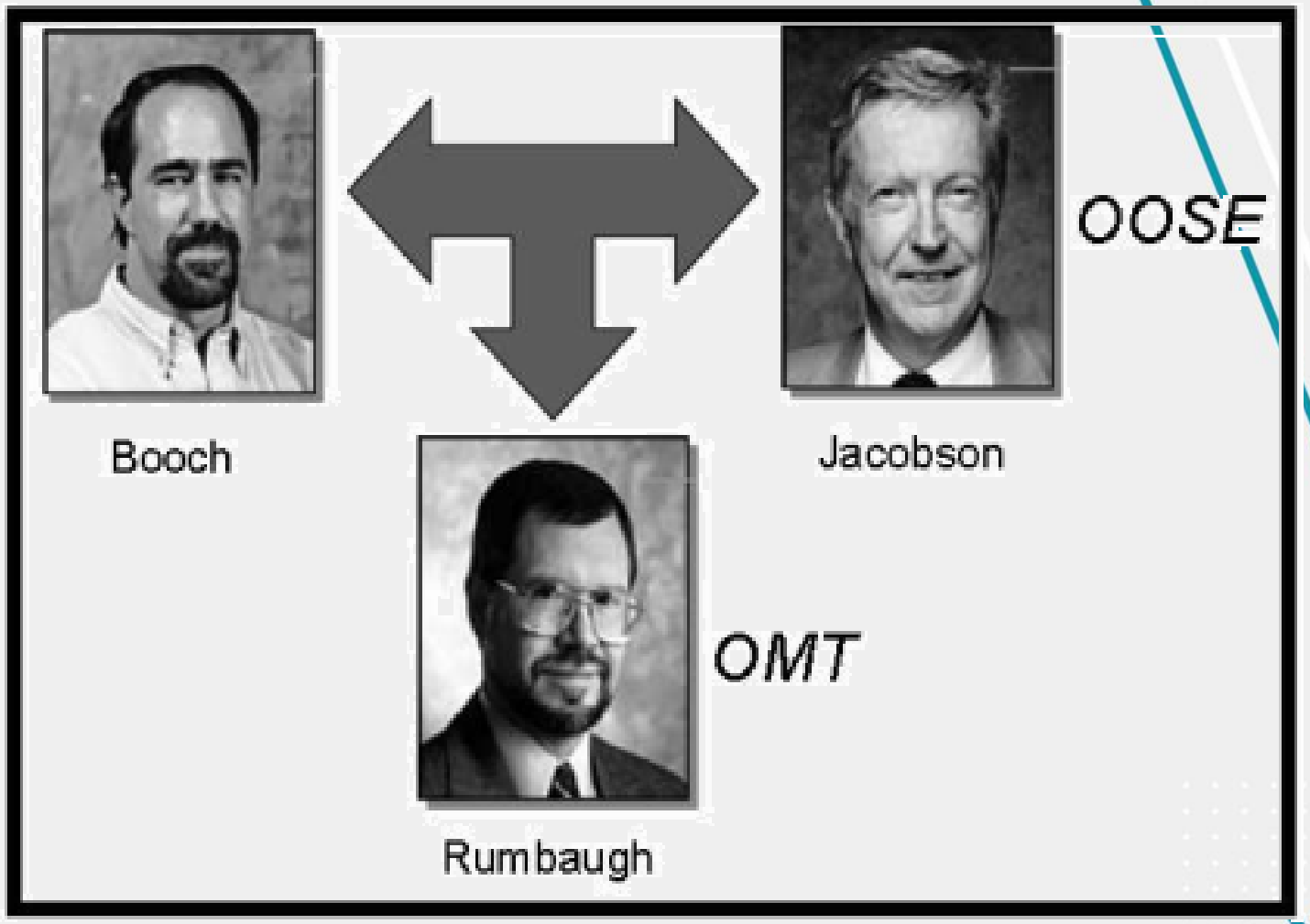
# UML
# Diagrama de Classes

UNISUAM




É uma linguagem de modelagem visual utilizada para especificar, visualizar, desenvolver e documentar sistemas de software. É uma ferramenta essencial na Engenharia de Software, pois ajuda a representar a estrutura e o comportamento do sistema.

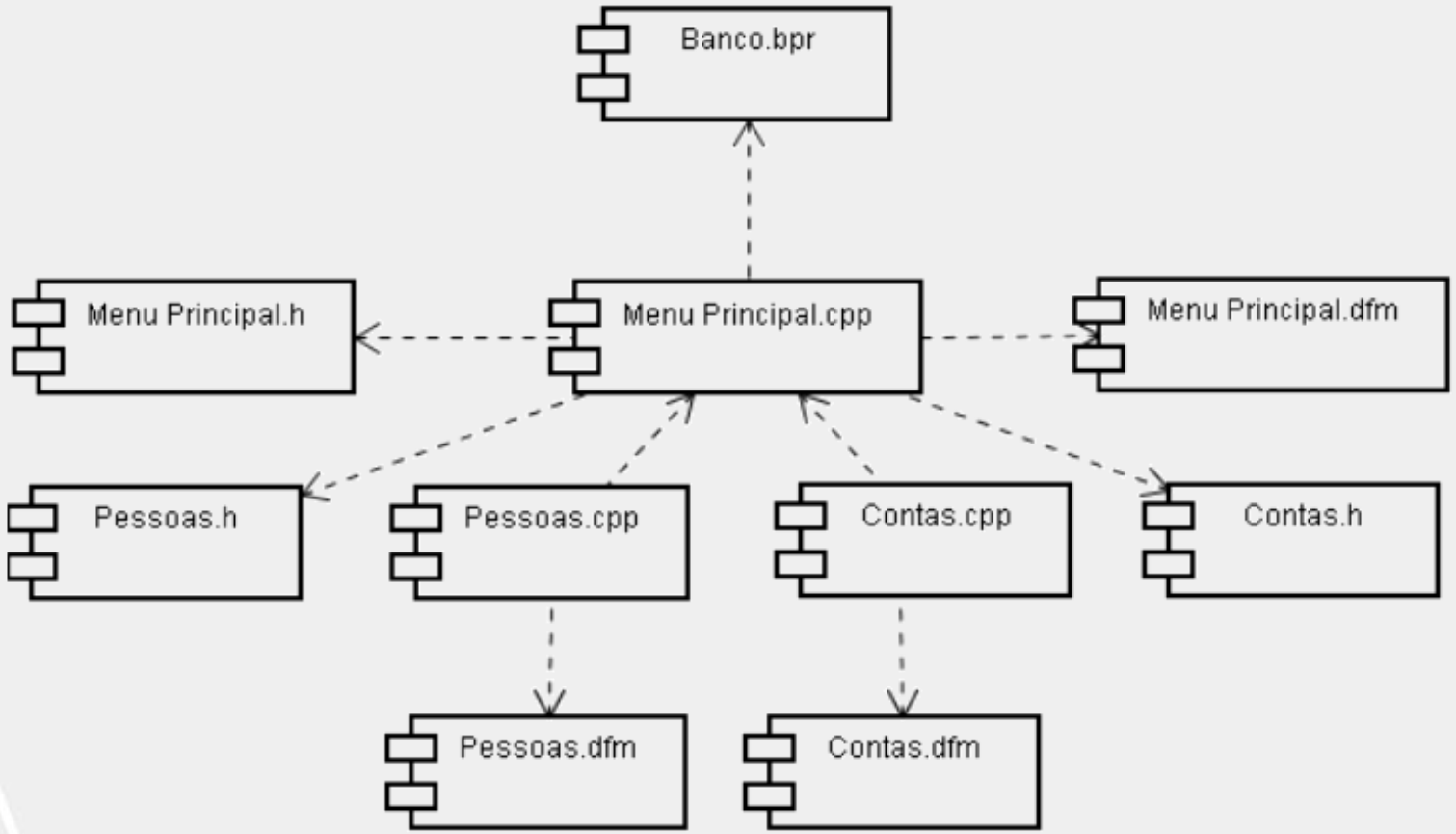
# Fundadores

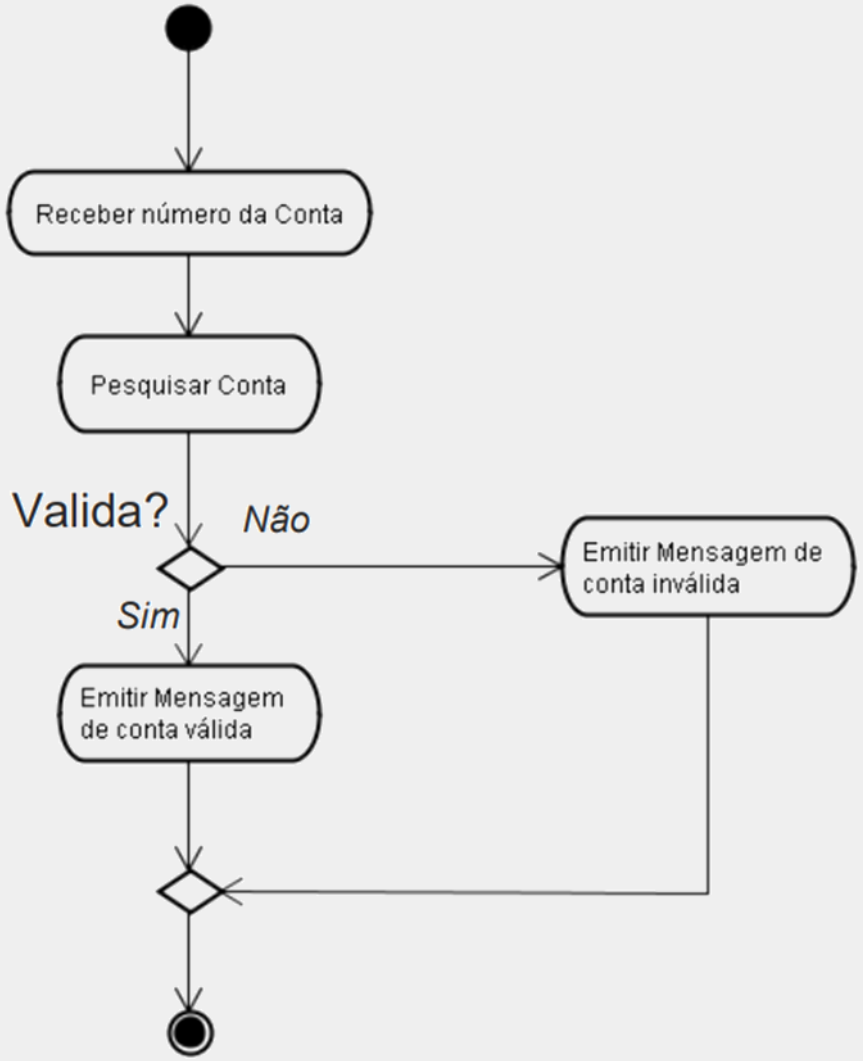
# Principais Diagramas

UNISUAM

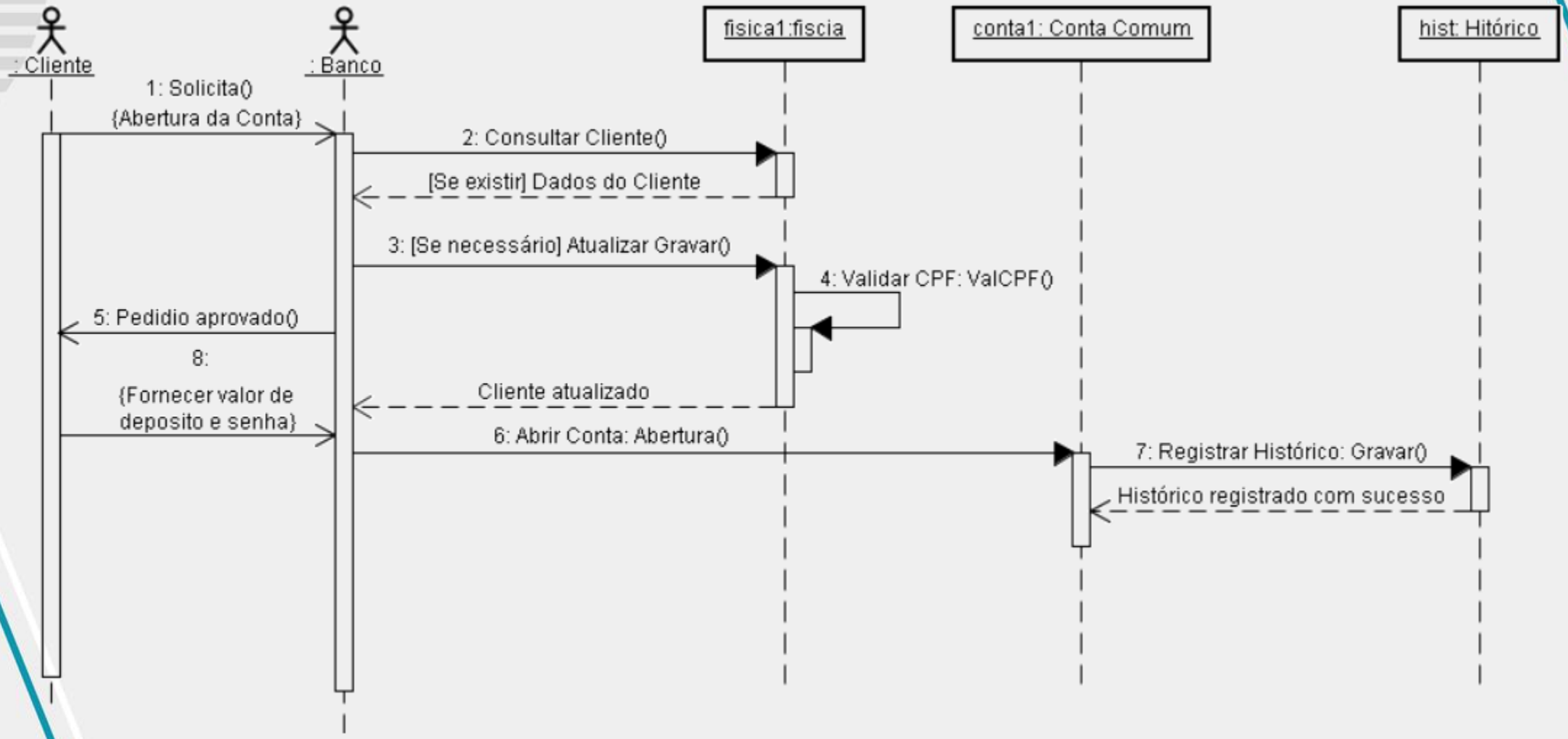
# Diagrama de Componentes



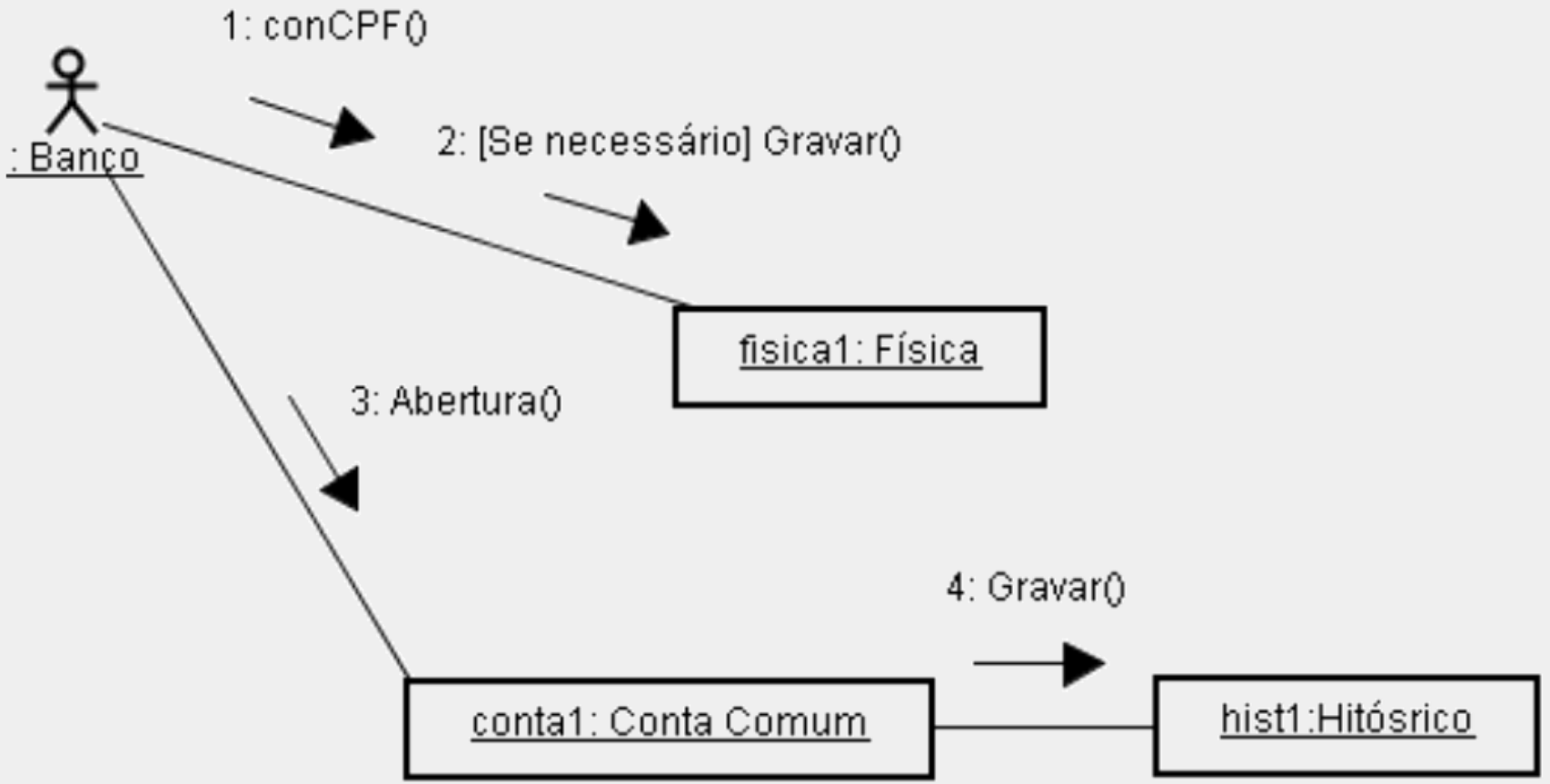
# Diagrama de Atividades



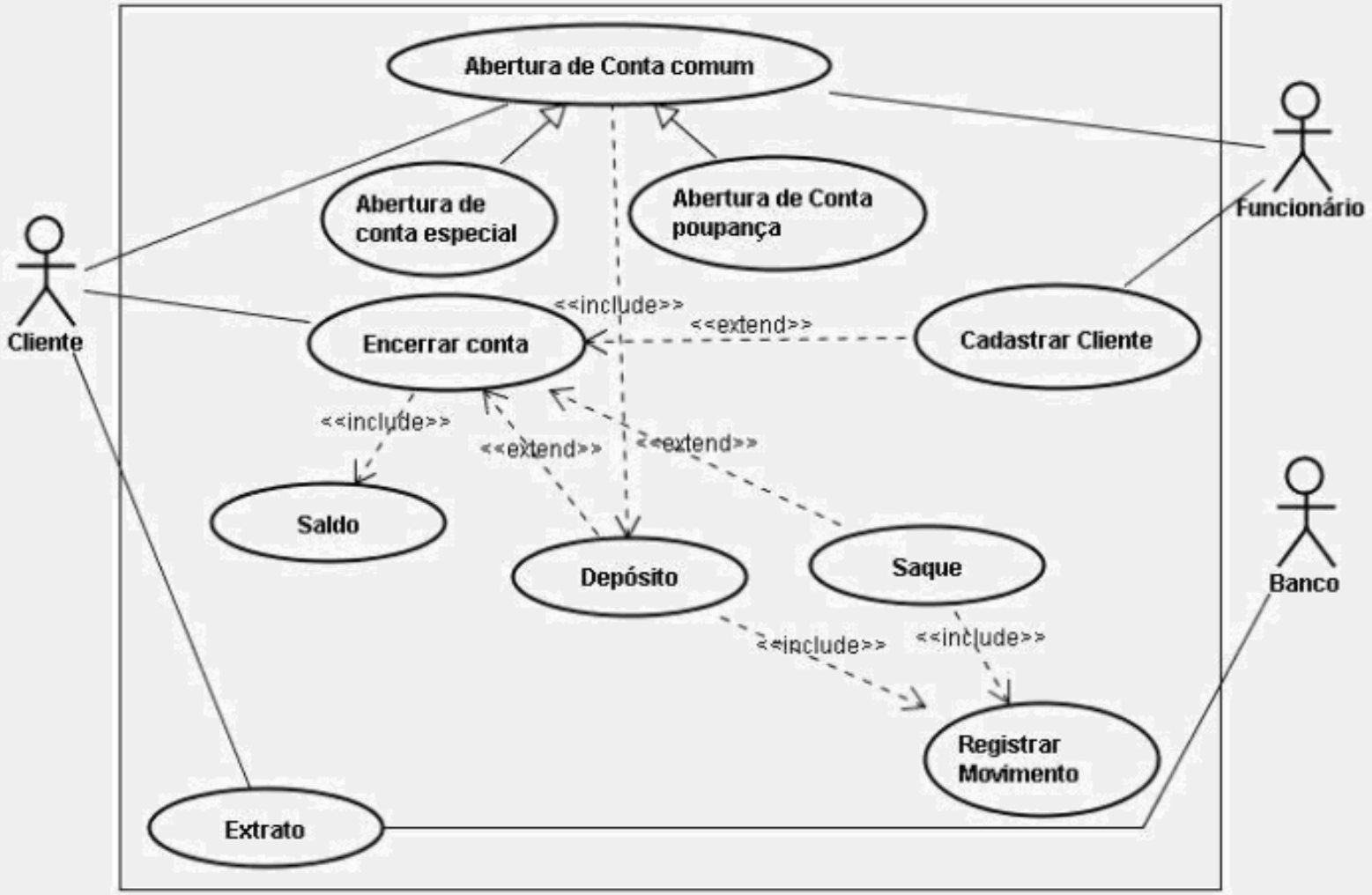
# Diagrama de Sequencia



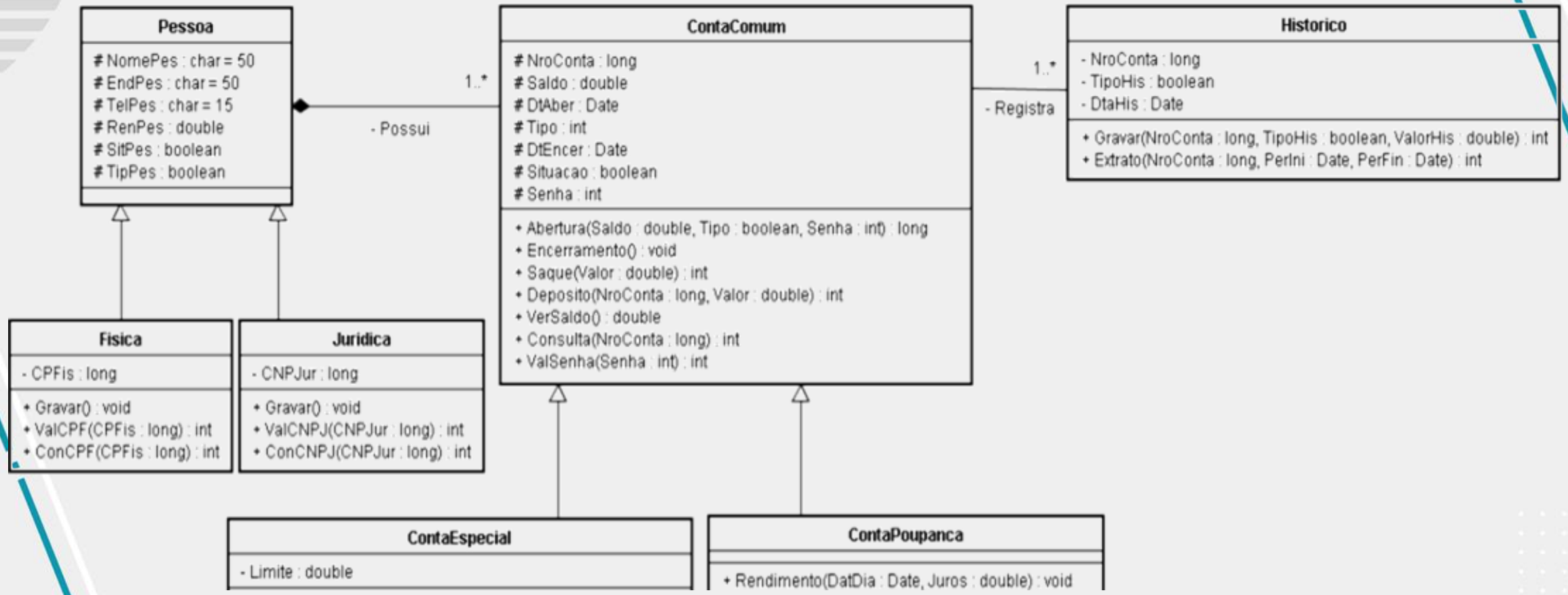
# Diagrama de Colaboração



# Diagrama de Caso de Uso



# Diagrama de Classes



# Diagrama de Classes

UNISUAM

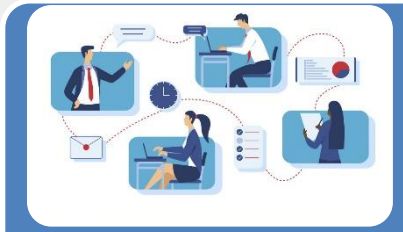
# A Programação Orientada a Objetos (POO) é um paradigma de desenvolvimento que modela o software com base em objetos, representando entidades do mundo real.

## Modelagem que Reflete o Mundo Real

- A POO permite modelar o software de uma forma natural, representando conceitos do mundo real, como pessoas, animais, plantas e veículos. Torna o código mais intuitivo, pois pensamos em termos de objetos e suas interações.

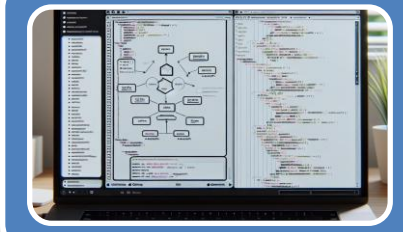
## Facilidade de Comunicação

- Bons modelos orientados a objetos facilitam a comunicação entre os stakeholders, garantindo que todos os envolvidos no projeto compreendam a estrutura e o funcionamento do sistema.



## Padronização e Manutenção

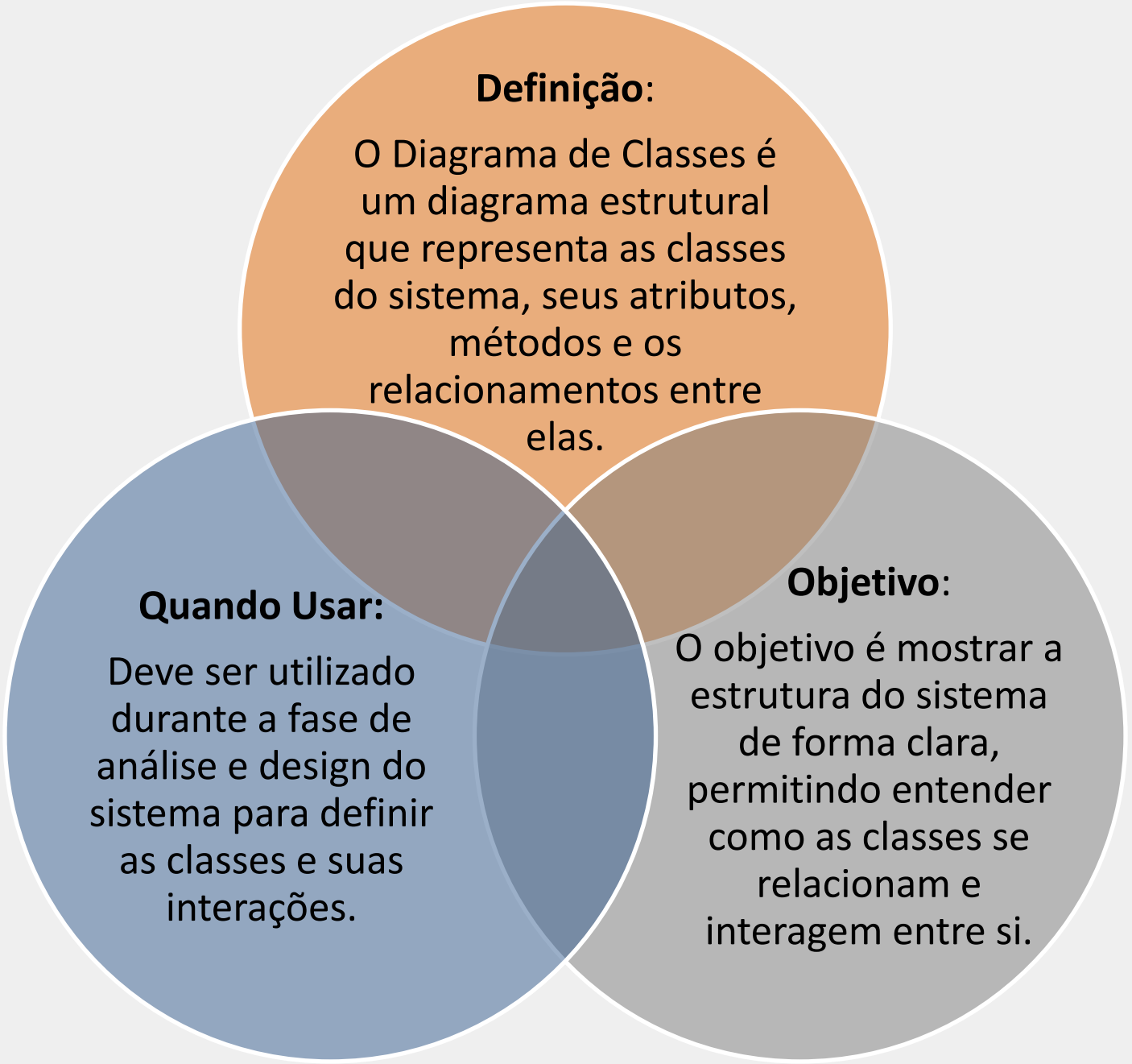
- Quando o time adota padrões de modelagem, todos entendem melhor o código, o que facilita a manutenção e a evolução do sistema ao longo do tempo.



## Ferramentas de Modelagem e Geração de Código

- A POO conta com diversas ferramentas que auxiliam na modelagem e até na geração automática de código, acelerando o desenvolvimento e reduzindo erros.

A **UML** representa graficamente sistemas orientados a objetos, e os Diagramas de Classes facilitam a visualização da estrutura do software segundo os princípios da Programação Orientada a Objetos.


**Definição**:
O Diagrama de Classes é um diagrama estrutural que representa as classes do sistema, seus atributos, métodos e os relacionamentos entre elas.
**Quando Usar:**
Deve ser utilizado durante a fase de análise e design do sistema para definir as classes e suas interações.
**Objetivo**:
O objetivo é mostrar a estrutura do sistema de forma clara, permitindo entender como as classes se relacionam e interagem entre si.

O diagrama de classes pode ser comparado a uma planta baixa de uma casa, onde cada classe representa um cômodo e suas características, como tamanho e funcionalidade.


Assim como a planta baixa define a disposição e a relação entre os espaços da casa, o diagrama de classes organiza as classes e suas interações em um sistema, facilitando a compreensão da estrutura do software.

O objetivo de ambos é fornecer uma visão clara e estruturada do que será construído, ajudando na comunicação entre os envolvidos no projeto.

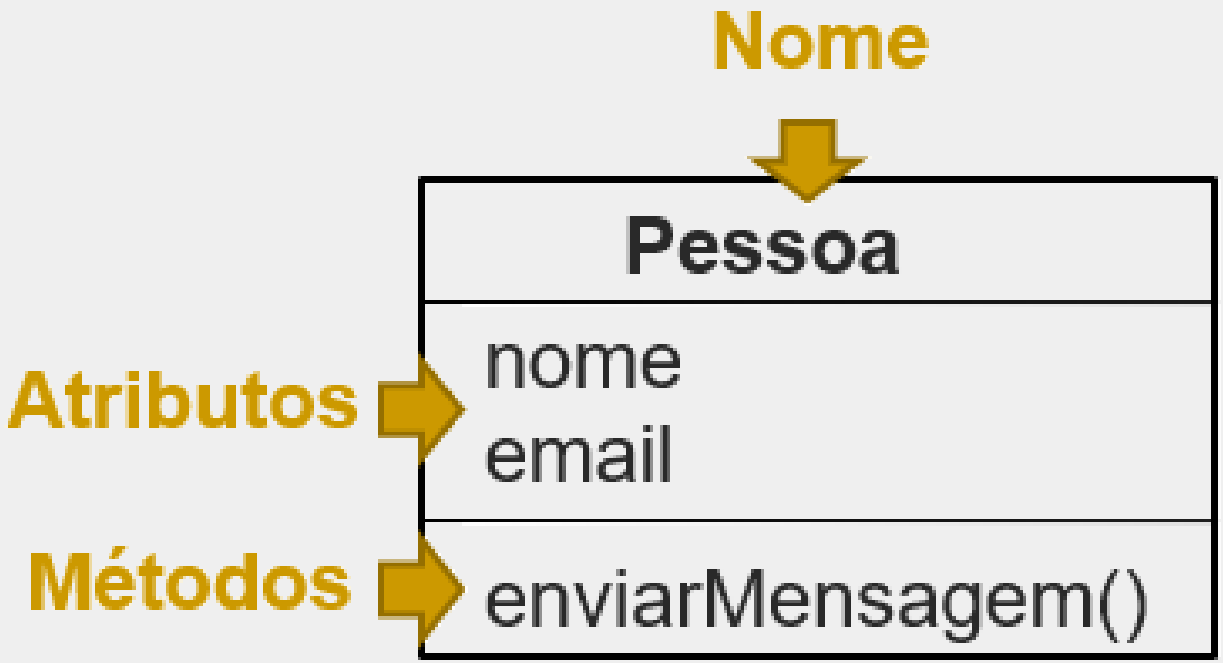
Utiliza-se o diagrama de classes no início do desenvolvimento, assim como a planta baixa é essencial antes da construção, garantindo que todos os elementos necessários sejam considerados desde o início.

# Componentes de um Diagrama de Classes

UNISUAM

# Classes

Representadas por retângulos que contêm o nome da classe, atributos e métodos.



# Visibilidade

## Publica (+)

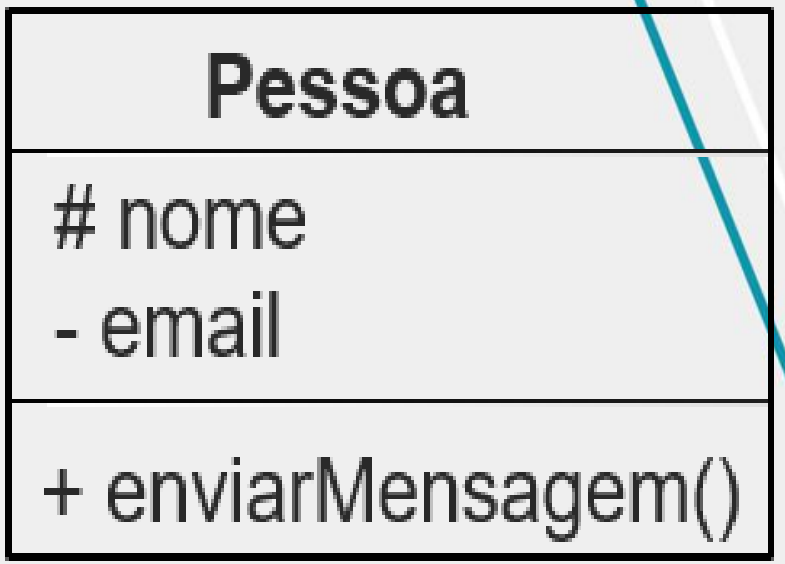
- Acessível por qualquer classe

## Privado(-)

- Acessível apenas dentro da própria classe.

## Protegido (#)

- Acessível pela própria classe e suas subclasses.



# Relacionamentos

Mostram como as classes interagem, eles são essenciais para modelar sistemas orientados a objetos.

Associação

Dependência

Agregação

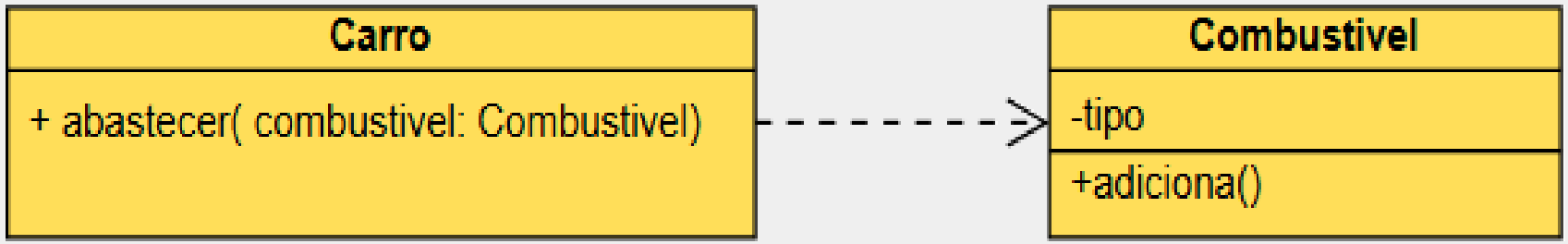
Composição

Herança

# Associação

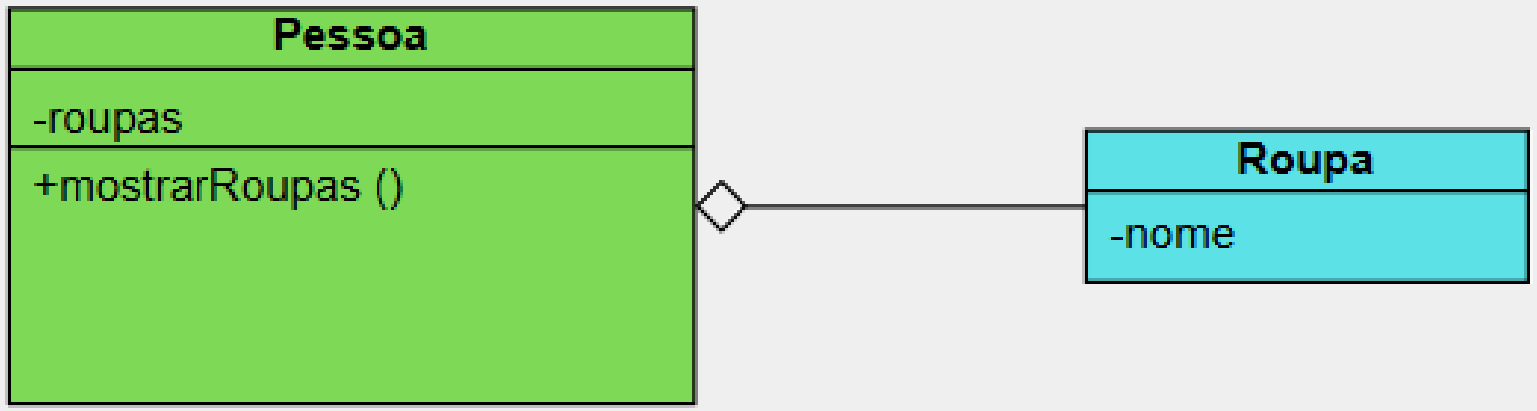

Indica que **uma propriedade de uma classe contém uma referência a uma instância (ou instâncias) de outra classe** .
Associação é o relacionamento **mais comumente usados** entre uma classe e uma classe, o que significa que há uma conexão entre um tipo de objeto e outro tipo de objeto. **Combinações e agregações também pertencem a relações associativas** , mas as relações entre classes de afiliações são mais fracas que as outras duas.
Existem quatro tipos de **associações** : **associações bidirecionais , associações unidirecionais , auto associação** e associações de **vários números** .

# Dependência - - - - - ->



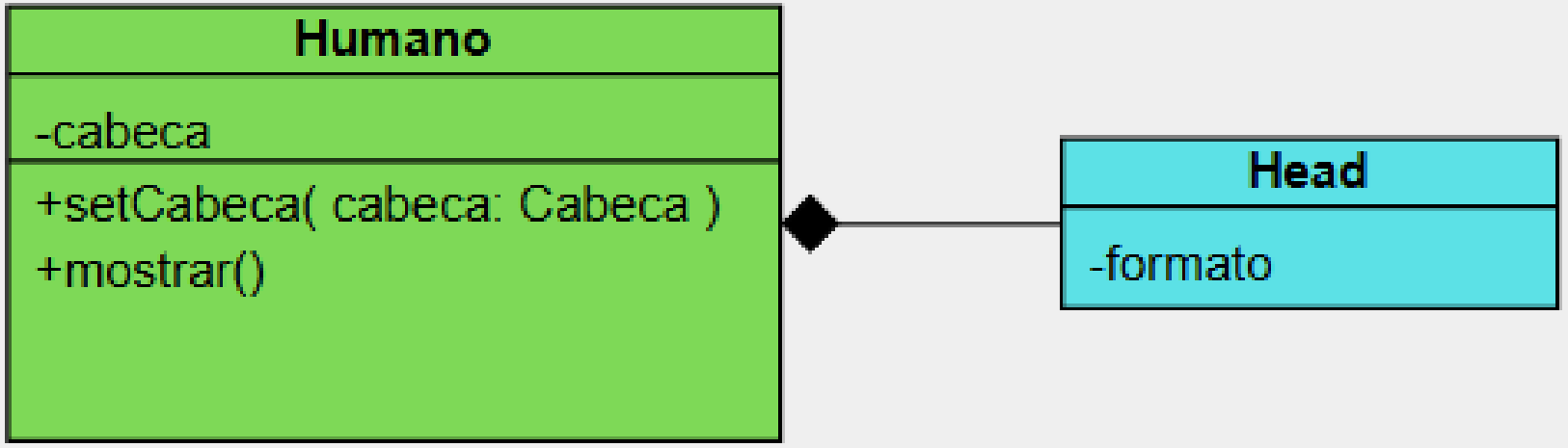

Dependência: Suponha que uma mudança na classe A cause uma mudança na classe B, então diga que a classe B depende da classe A.
Na maioria dos casos, **as dependências são refletidas em métodos de uma classe que usam o objeto de outra classe como parâmetro** .
Uma relação de dependência é uma relação de “uso”. Uma mudança em uma determinada coisa pode afetar outras coisas que a utilizam, e usar uma dependência quando é necessário indicar que uma coisa usa outra.

# Agregação



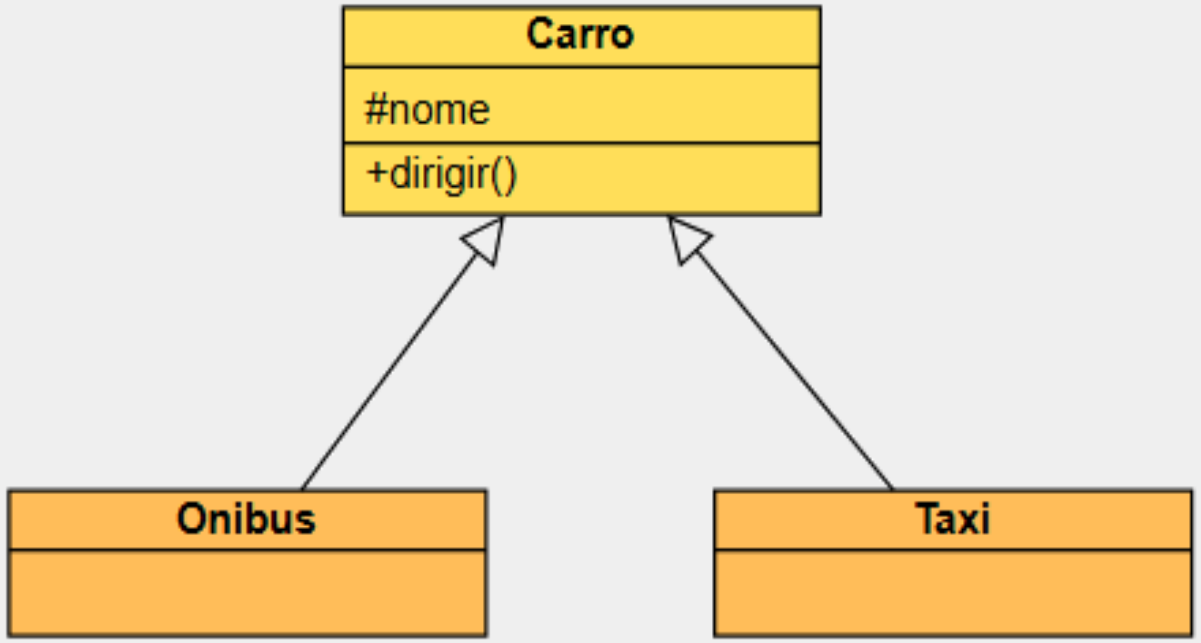

**A relação entre o todo e a parte, e o todo e a parte podem ser separados.** As relações agregadas também representam o relacionamento entre o todo e parte da classe, os objetos membros fazem parte do objeto geral, mas o objeto membro pode existir independentemente do objeto geral.

# Composição



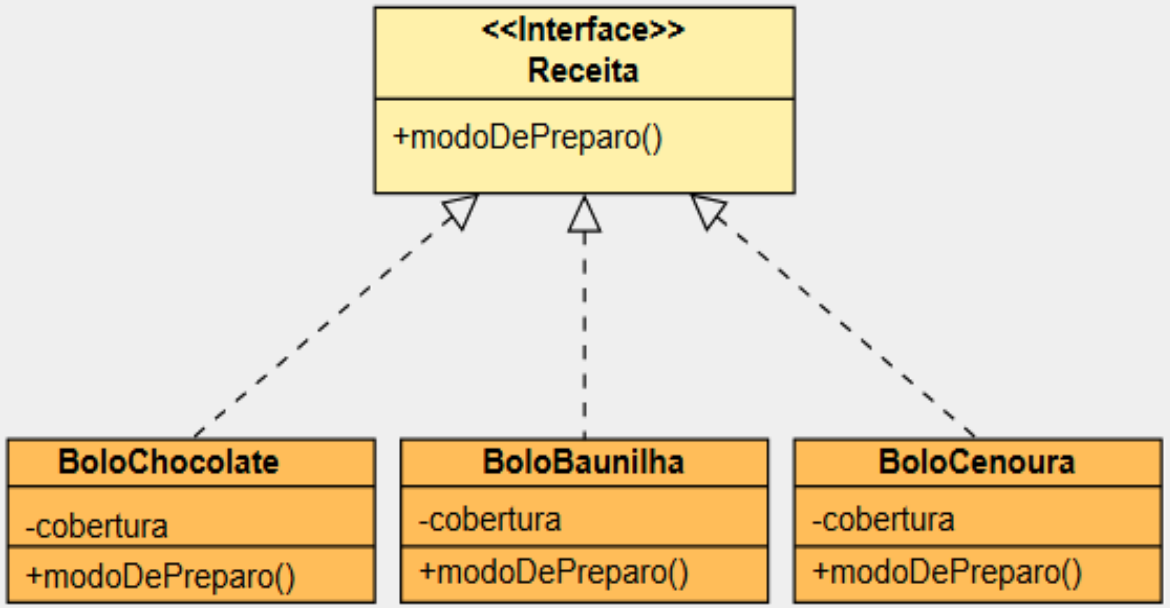

**A relação entre o todo e a parte, mas o todo e a parte não podem ser separados** .
A relação de combinação representa a relação entre o todo e parte da classe, e o geral e a parte têm um tempo de vida consistente. Uma vez que o objeto geral não exista, alguns dos objetos não existirão e todos morrerão na mesma vida.

# Herança




**A herança** também é chamada de **generalização** e é usada para descrever o relacionamento entre as classes pai e filha. Uma classe pai também é chamada de classe base e uma subclasse também é chamada de classe derivada.
No relacionamento de herança, a subclasse herda todas as funções da classe pai, e a classe pai tem todos os atributos, métodos e subclasses. As subclasses contêm informações adicionais além das mesmas informações que a classe pai.

# Realização/Implementação (Interface)

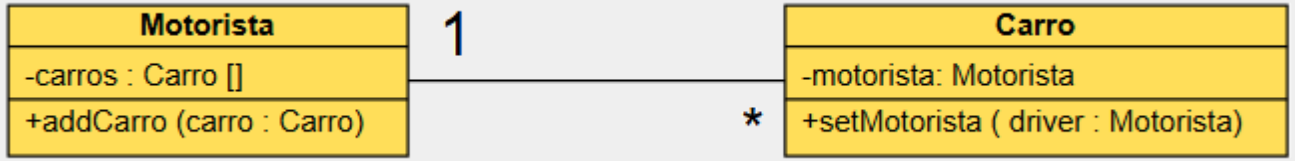



**Implementação** (Implementação) é usada principalmente para especificar **o relacionamento entre interfaces e classes de implementação** .
**Uma interface** (incluindo uma **classe abstrata** ) é uma coleção de métodos. Em um relacionamento de implementação, uma classe implementa uma interface e os métodos da classe implementam todos os métodos da declaração da interface.

# Multiplicidade

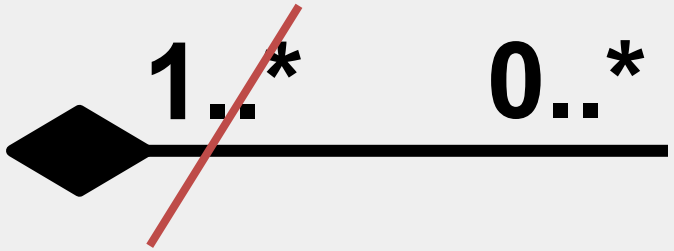
Especifica o número mínimo e máximo de instâncias que podem ser vinculadas.

| | |
|---|---|
| 1 | Exatamente um, nem mais nem menos |
| 0..1 | Zero ou um |
| * | Muitos |
| 0..* | Zero ou muitos |
| 1..* | um ou muitos |




- Os valores mínimo e máximo são separados por dois pontos.
- Um asterisco (*) representa um valor máximo indeterminado.
- Caso os valores mínimo e máximo sejam iguais, o número é exibido apenas uma vez.
- Além disso, zero ou mais (0... *) também é chamado de asterisco simples (*) porque é comum.
- Quando não vemos a multiplicidade representa uma única instância.

# OBSERVAÇÃO




Errado, pois quando o “todo” morre todas as suas “partes” também morrem

**A composição é um relacionamento mais forte: a parte não pode existir sem o todo.**

**Exemplo:**

Um **Pedido** contém **Itens**, e se o pedido for destruído, os itens deixam de existir.

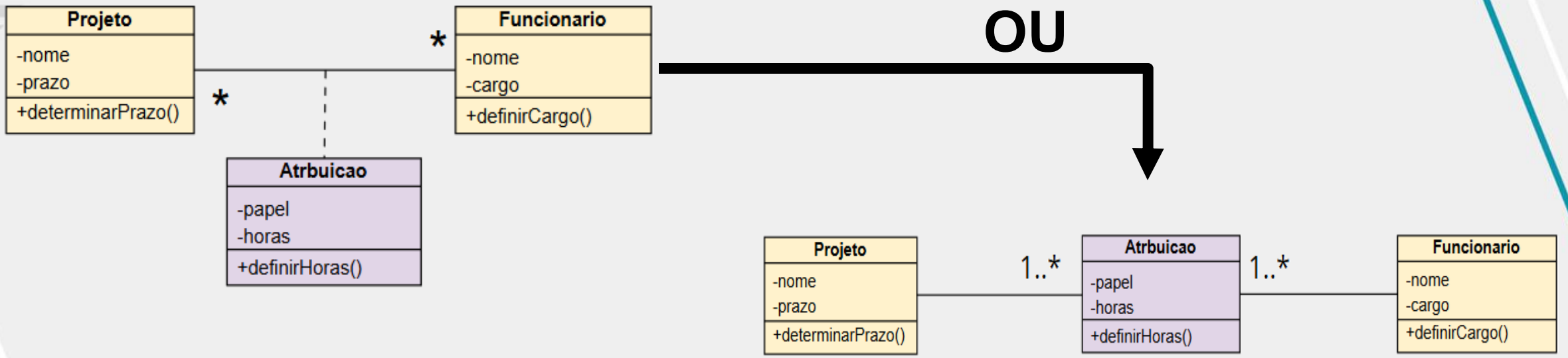
**Normalmente um todo contém muitas partes, ou seja, multiplicidade 1 para muitos, mas também pode ser 1 para 1.**

**Muitos-para-muitos**

Não é conceitualmente coerente ter uma multiplicidade muitos-para-muitos em composição.
Como as partes dependem da existência do todo, um mesmo objeto (parte) não pode pertencer simultaneamente a vários "todos".

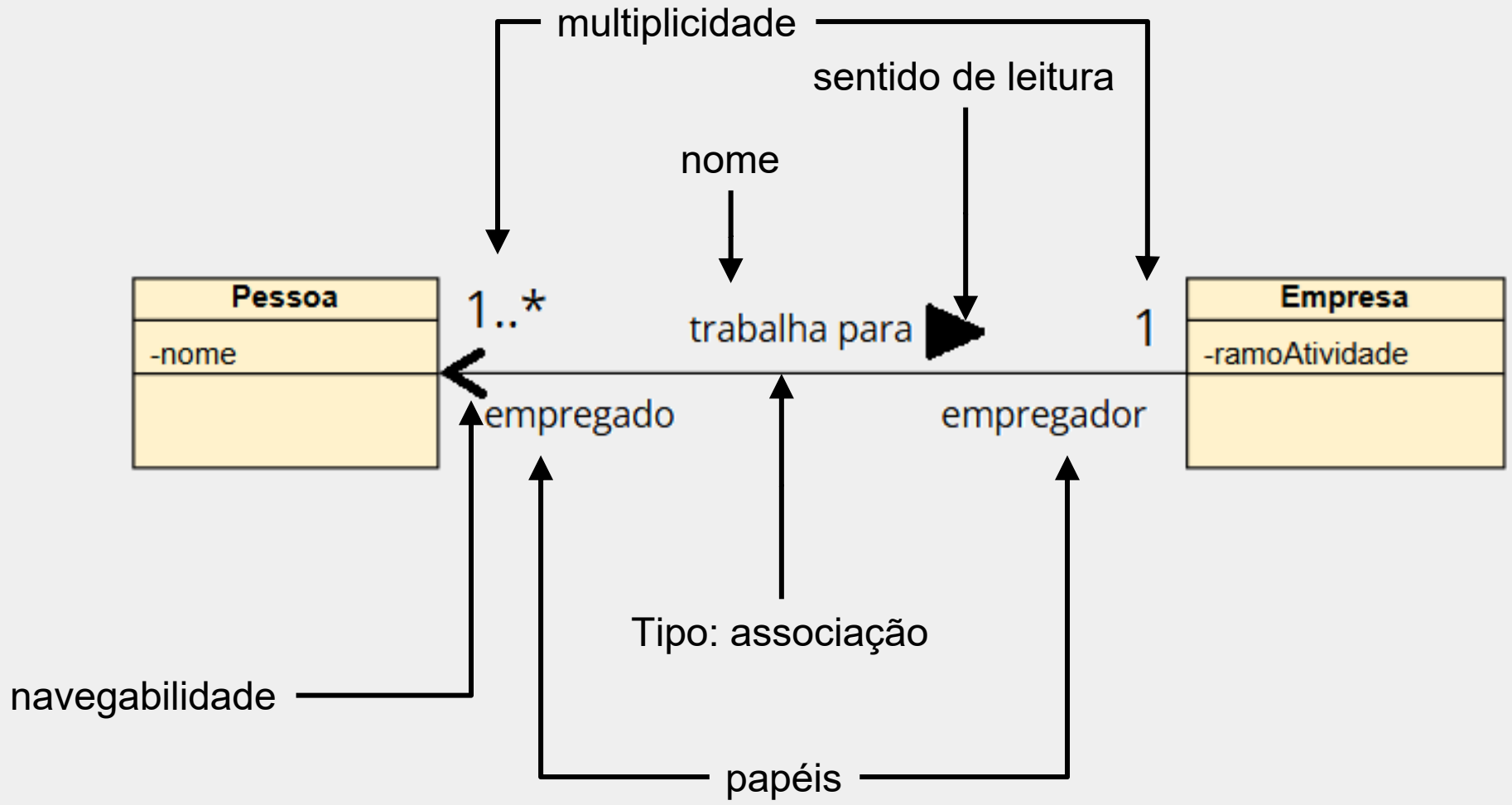
# Classe Associativa

É uma classe ligada a uma associação, com nome, atributos e operações próprios




Uma classe de associação, que é essencialmente uma classe anexada a uma associação, é usada para modelar uma associação como uma classe UML . Ela tem seu próprio nome, operações de atributos, assim como qualquer outra classe comum. No entanto, ela é descrita por atributos adicionais que não pertencem aos objetos envolvidos na associação.

# Componentes Completos de Relacionamento em Diagramas de Classes UML



# Dicas de Links:


Curso de UML O que é um Diagrama de Classes

Bóson Treinamentos • 184 mil visualizações • há 6 anos

Curso de #UML - O que é um Diagrama de Classes Neste vídeo apresentamos o Diagrama de Classes UML Seja membro deste canal e ganhe benefícios: https://www.youtube.com/channel/UCzOGJclZQvPVgYZIw...

https://www.youtube.com/watch?v=JQSsqMCVi1k


Curso de UML - Diagrama de Classes - Relacionamentos

Bóson Treinamentos • 168 mil visualizações • há 6 anos

Curso de #UML - Diagrama de Classes - Relacionamentos Seja membro deste canal e ganhe benefícios: https://www.youtube.com/channel/UCzOGJclZQvPVgYZIwERsf5g/join Contribua com a Bóson Treinamento...

https://www.youtube.com/watch?v=IJtQWLnHvcQ

# DÚVIDAS?

Profª: Kellen Nery
Kellennery@souunisuam.com.br

UNISUAM