

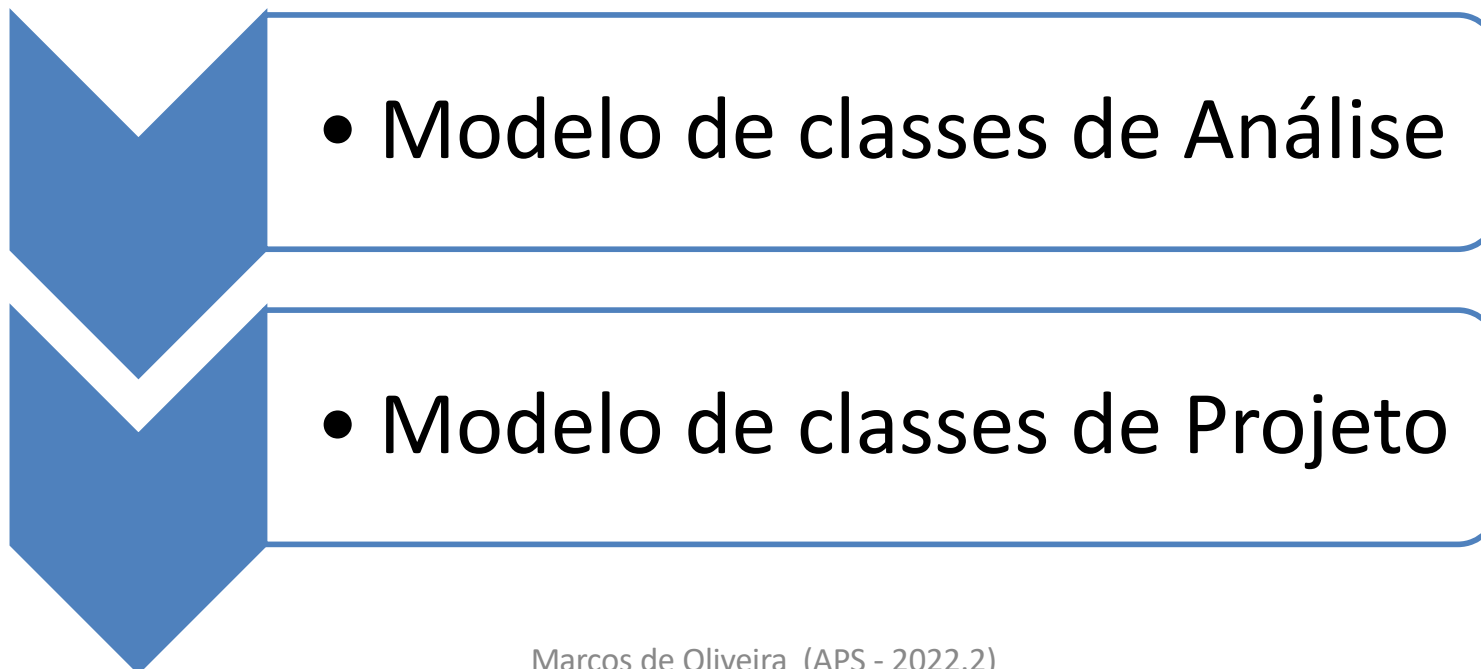
# Análise e Projeto de Sistemas

Universidade Federal do Ceará – UFC  
Campus de Quixadá  
Curso de Sistemas de Informação  
Prof. Marcos de Oliveira

Esses slides são uma adaptação das notas de aula do professor Eduardo Bezerra autor do livro Princípios de Análise e Projeto de Sistemas com UML

# Modelo de Classes

- O diagrama da UML utilizado para representar o aspecto estático é o **diagrama de classes**
- É detalhado ao longo do desenvolvimento



# DIAGRAMA DE CLASSES

# Notação para uma Classe

- Representada através de uma “caixa” com no máximo três compartimentos exibidos
- Notação utilizada depende do nível de abstração desejado

<b>Nome da Classe</b>
-----------------------

<b>Nome da Classe</b>
lista de atributos

<b>Nome da Classe</b>
lista de operações

<b>Nome da Classe</b>
lista de atributos
lista de operações

# Exemplo (classe ContaBancária)

<b>ContaBancária</b>
----------------------

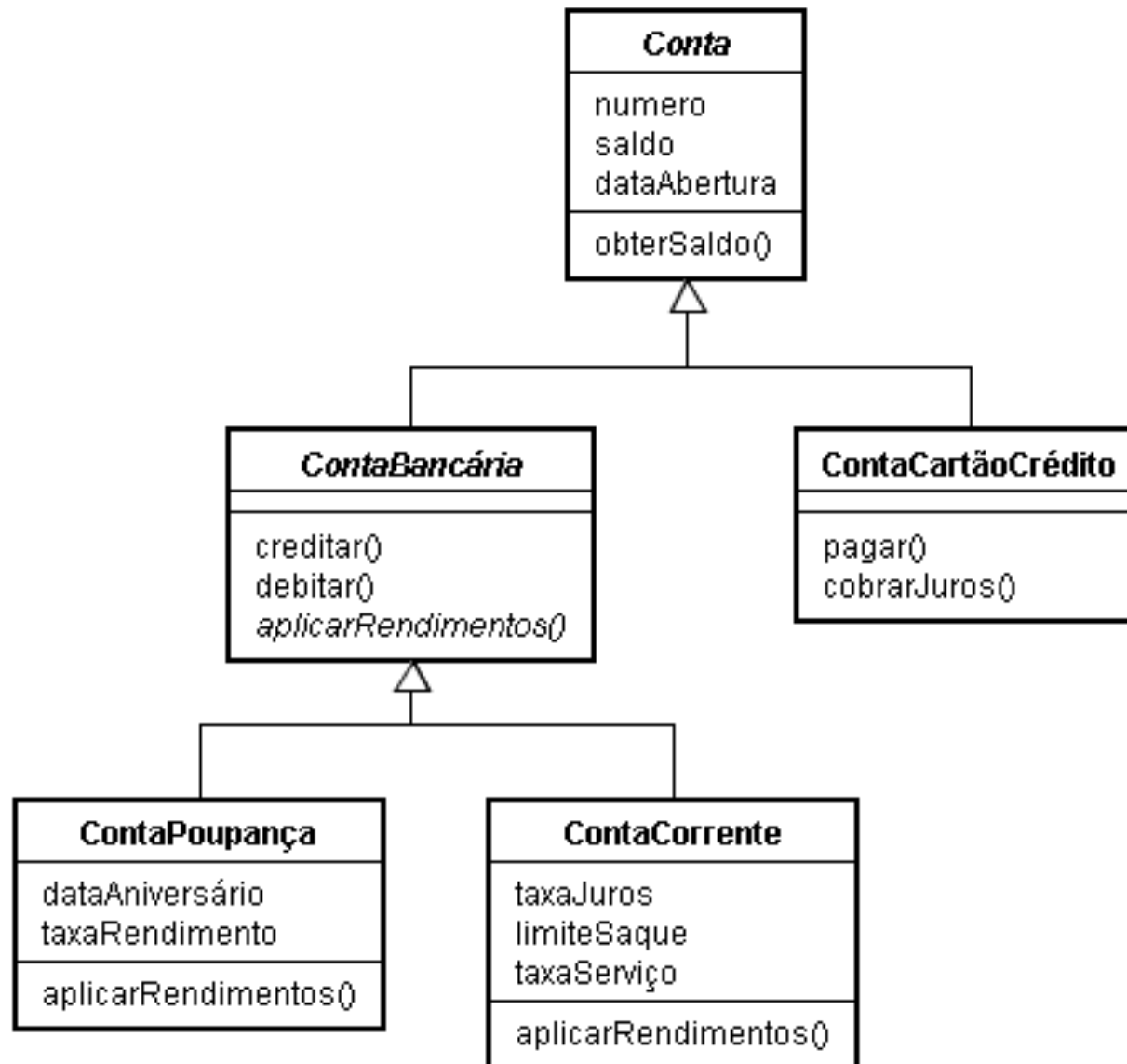
<b>ContaBancária</b>
número saldo dataAbertura

<b>ContaBancária</b>
número saldo dataAbertura
criar() bloquear() desbloquear() creditar() debitar()

# Relacionamentos

- Generalização/Especialização (Herança)
- Associação

# Exemplo (Herança)



# Herança

- As principais propriedades do relacionamento de herança são
  - *Transitividade*
    - Uma classe herda tanto os **atributos**, **operações** e **relacionamentos** da super classe imediata quanto das superclasses **não imediatas**
  - *Assimetria*
    - Dadas duas classes A e B, se B é subclasse de A, então A não pode ser subclasse de B

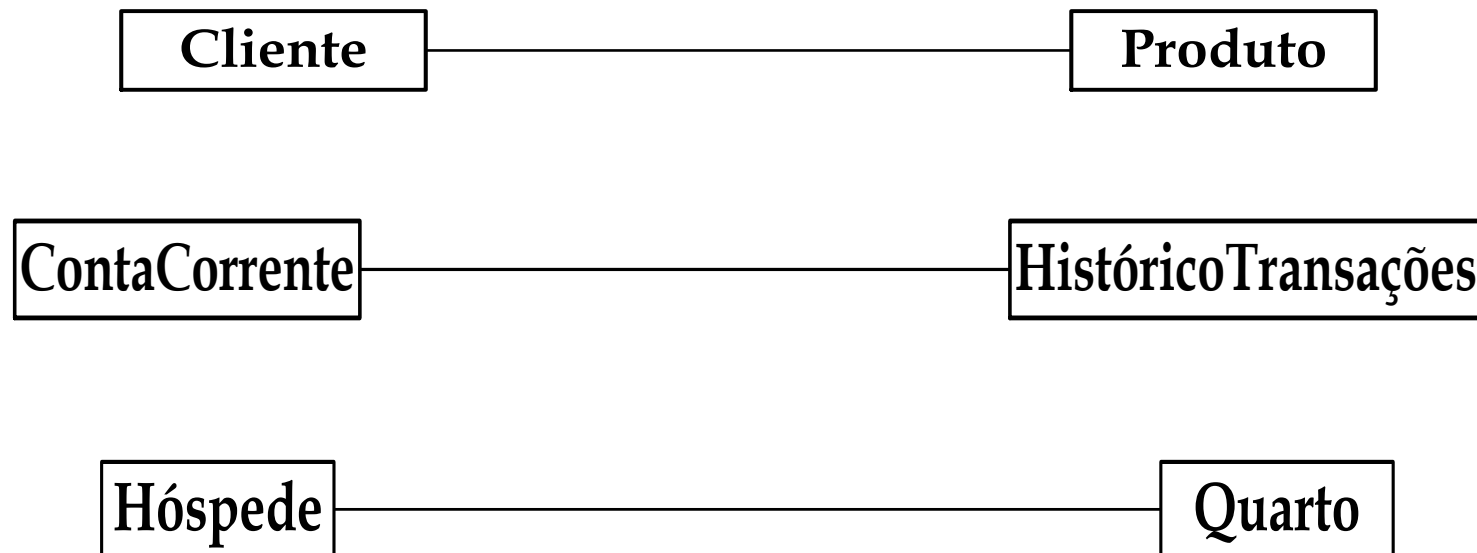


# Associações

- Para representar o fato de que classes podem se relacionar umas com as outras
- Uma associação representa relacionamentos (ligações) que são formados entre objetos durante a execução do sistema

# Notação para Associações

- Representada através de um segmento de reta ligando as **classes** cujos **objetos** se relacionam



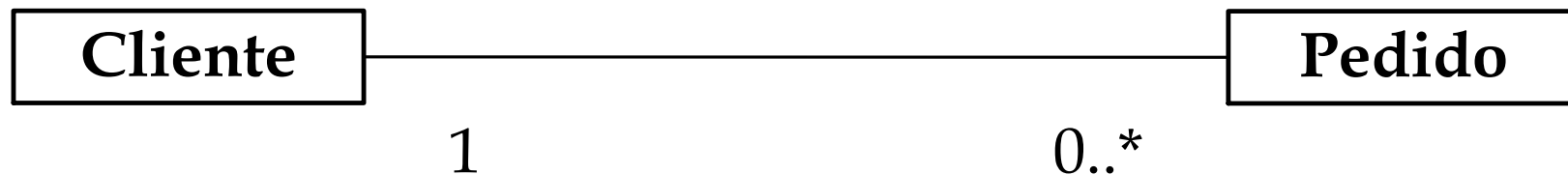
# Multiplicidade

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

Nome	Simbologia
Apenas Um	1..1 (ou 1)
Zero ou Muitos	0..* (ou *)
Um ou Muitos	1..*
Zero ou Um	0..1
Intervalo Específico	$l_i..l_s$

# Exemplo (Multiplicidade)

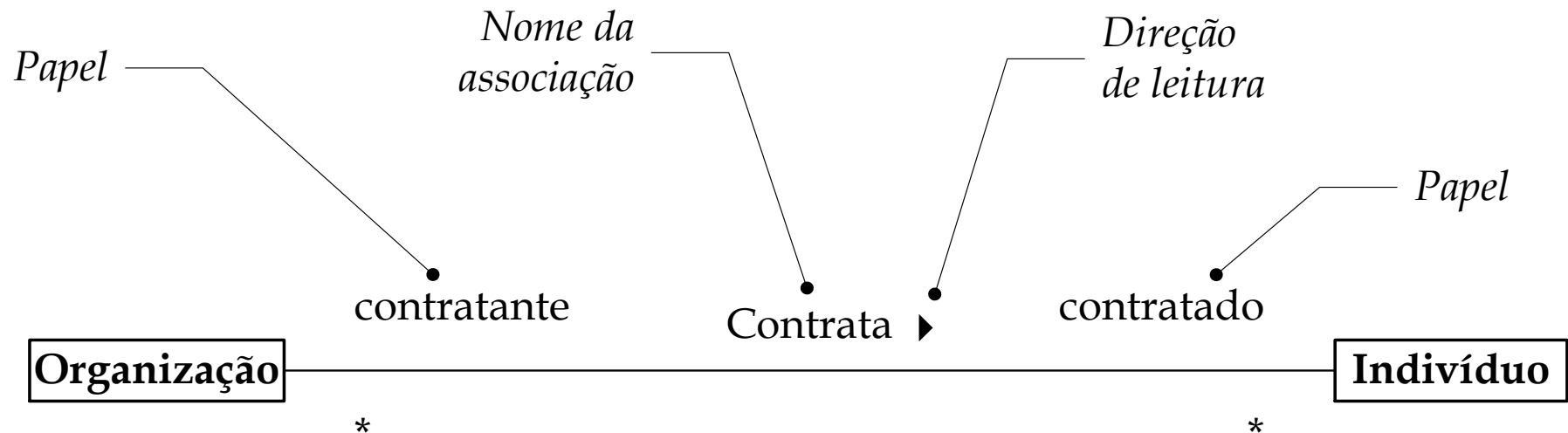
- **Pode** haver um cliente que esteja associado a vários pedidos
- **Pode** haver um cliente que não esteja associado a pedido algum
- Um pedido está associado a um, e somente um, cliente



# Nome de Associação, Direção de Leitura e Papéis

- **Nome da associação:** fornece algum significado semântico a mesma
- **Direção de leitura:** indica como a associação deve ser lida
- **Papel:** para representar um papel específico em uma associação

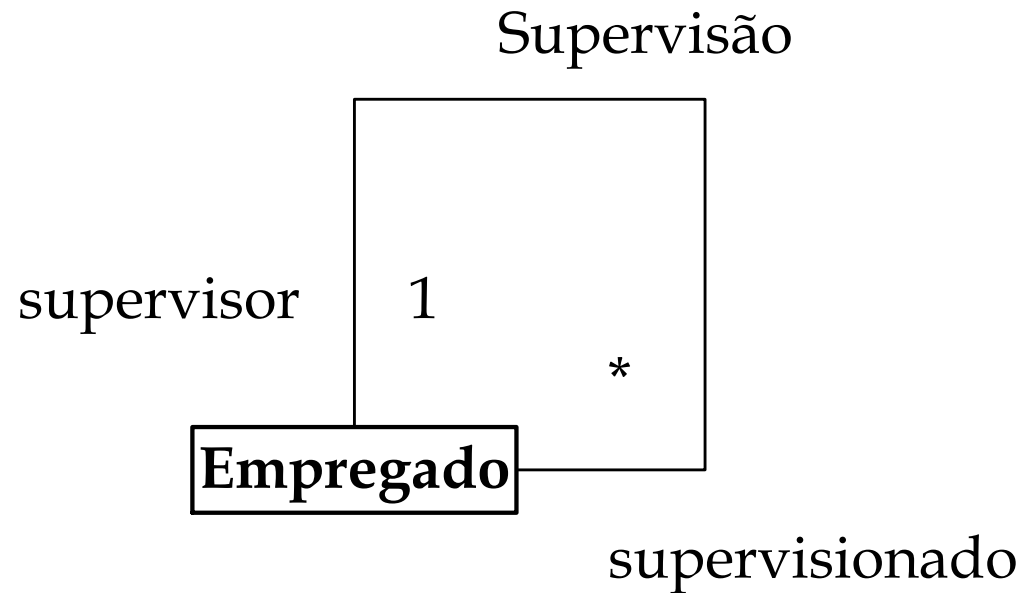
# Exemplo (Nome de Associação, Direção de Leitura e Papéis)



# Associações Reflexivas

- Associa objetos da mesma classe
- Cada objeto tem um papel distinto na associação
- A utilização de papéis é bastante importante para evitar ambigüidades na leitura da associação
- Uma associação reflexiva **não** indica que um objeto se associa com ele próprio
  - Ao contrário, indica que objetos de uma mesma classe se associam

# Exemplo (Associação Reflexiva)



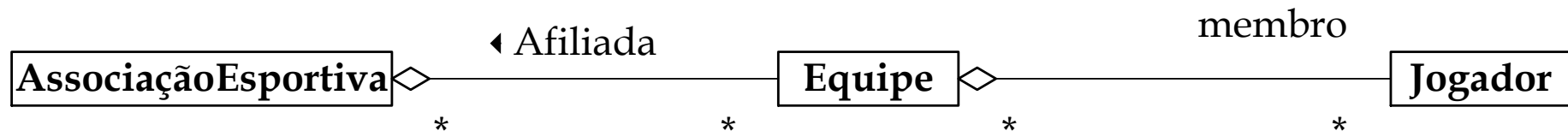


# Agregações e Composições

- É um caso especial da associação...  
...conseqüentemente, multiplicidades, participações, papéis, etc. podem ser usados igualmente
- Utilizada para representar conexões que guardam uma relação todo-parte entre si
- São assimétricas: se um objeto A é parte de um objeto B, B não pode ser parte de A

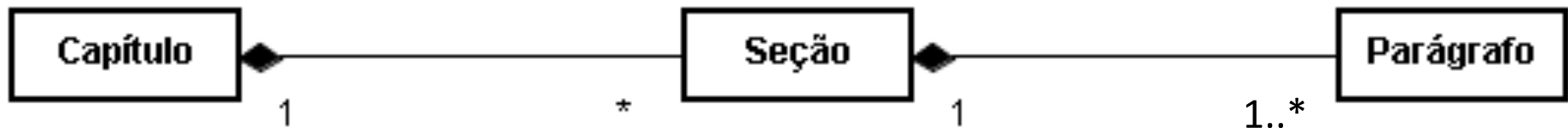
# Notação para Agregação

- Uma agregação é representada como uma linha que conecta as classes relacionadas, com um diamante (losango) **branco** perto da classe que representa o todo



# Notação para Composição

- Uma composição é representada como uma linha que conecta as classes relacionadas, com um diamante (losango) **negro** perto da classe que representa o todo



# Agregações vs Composições

- As diferenças entre agregações e composições não são bem claras, mas essas são as mais visíveis
  - Na agregação, a destruição de um objeto **todo** não implica necessariamente a destruição do objeto **parte**
  - Na composição, os objetos **parte** pertencem a único objeto **todo**, por isso quando um objeto **todo** é destruído os objetos **parte** também são

# Outros elementos

- Dependência – Indica que uma classe usa informações de outra

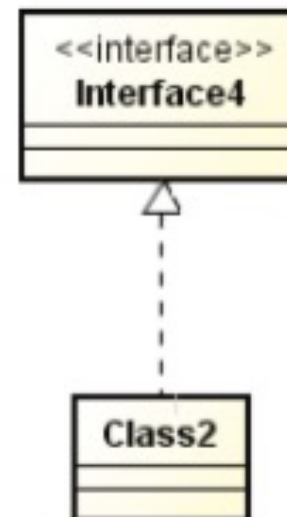


# Outros elementos

- Interface = Interface de código

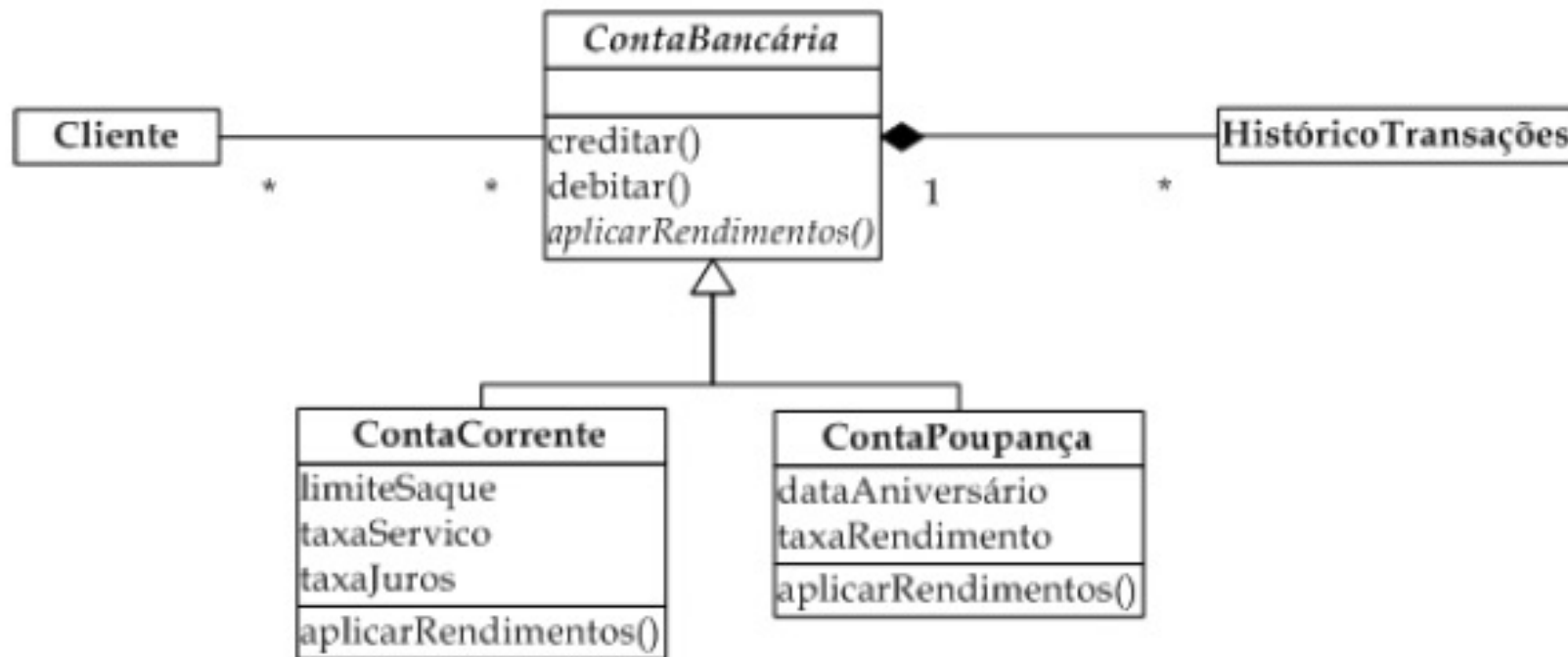


- Realização = generalização/especialização entre interface e classe

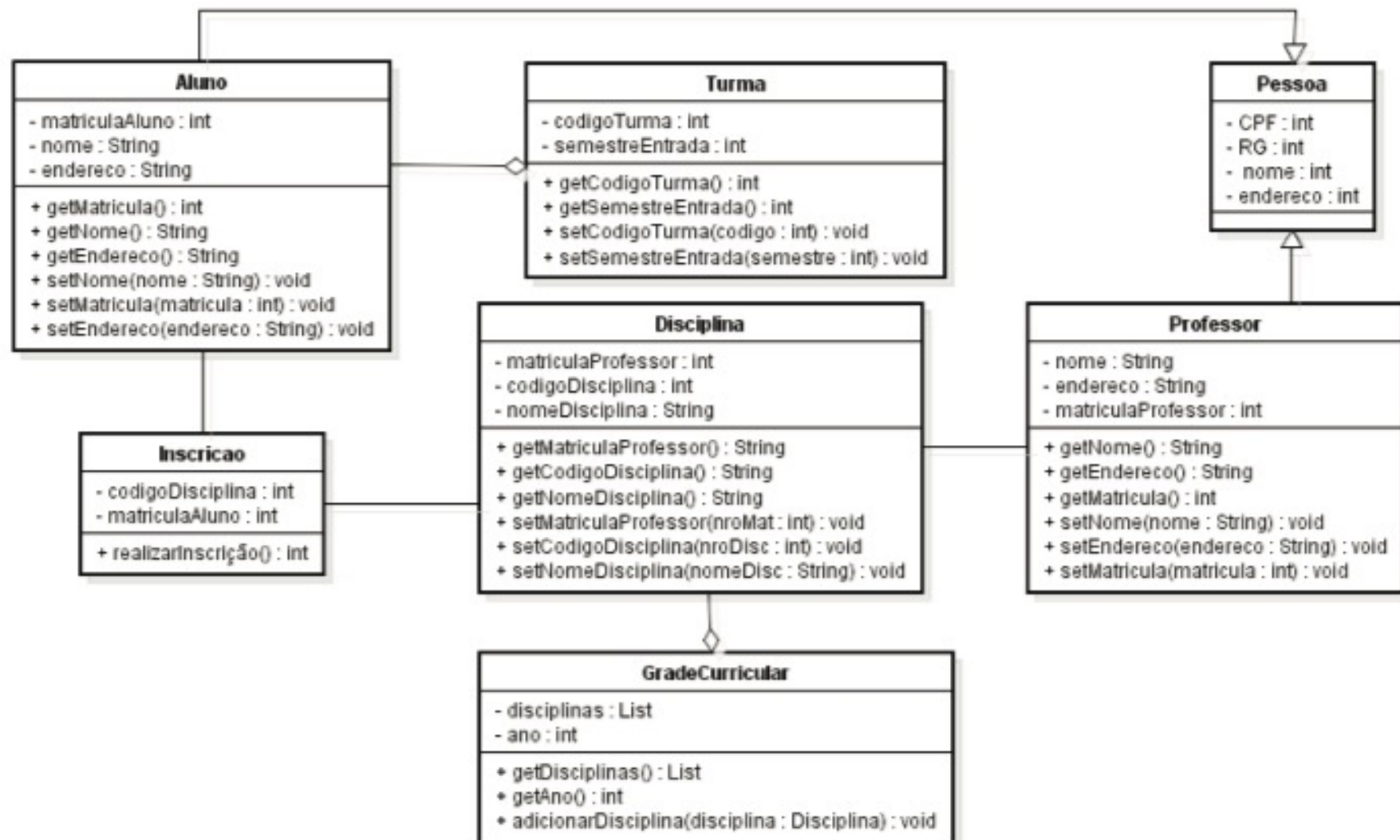


# Notação para Classe Abstrata

- Na UML, uma classe abstrata é representada com o seu nome em **itálico**



# Exemplo





# Referências

- BEZERRA, E. Princípios de Análise e Projeto de Sistemas com UML. 2ª ed. Rio de Janeiro: Elsevier, 2007.
- FOWLER, M. 3. UML Essencial. 3. ed. Porto Alegre: Bookman, 2007.