Diagrama de classes

Prof. Murillo G. Carneiro
FACOM/UFU

Material baseado nos slides disponibilizados pelo Prof. Ricardo Pereira e Silva (UFSC)

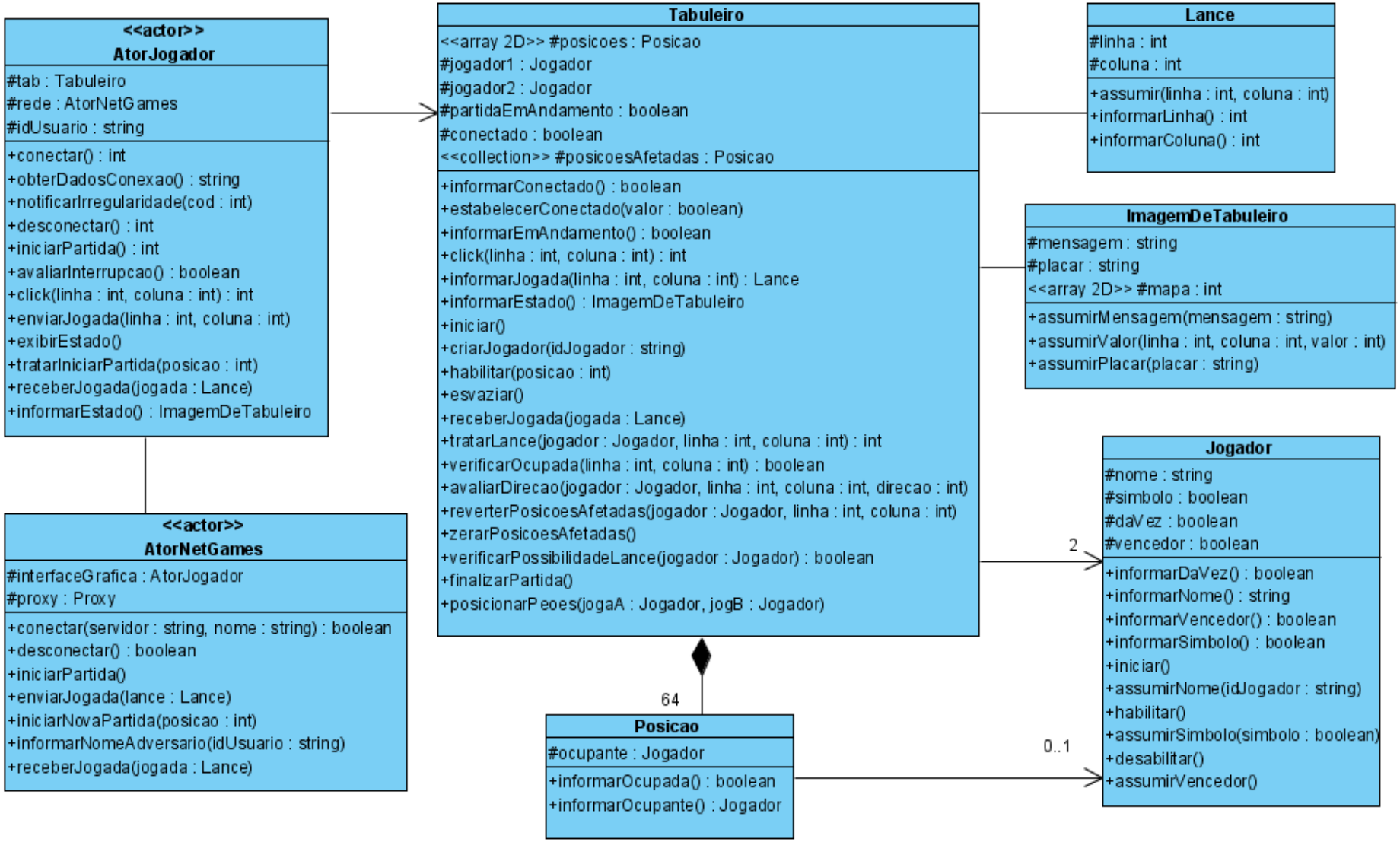
Objetivo

- Apresentar o diagrama de classes de UML
  - Seus elementos sintáticos
  - Sua finalidade em um processo de modelagem

Diagrama de classes

- Resultado do esforço mais primitivo de geração de modelos gráficos que descrevessem um programa orientado a objetos
- Reflete a estrutura do código
  - Classes
    - Cada classe com seus atributos e métodos
  - Relacionamentos envolvendo classes

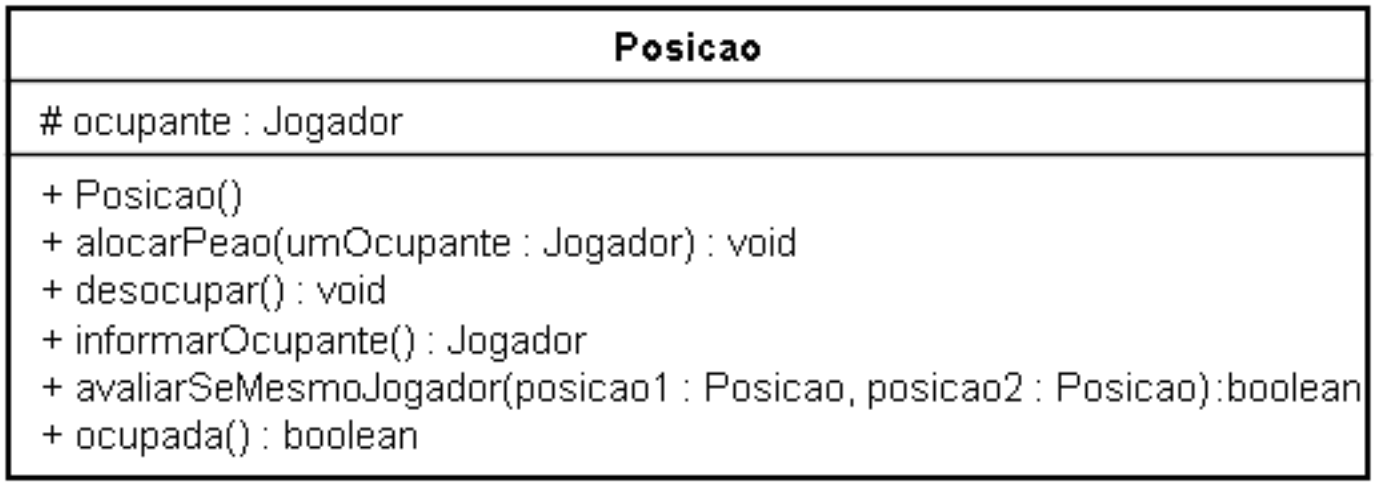
Aparência do diagrama de classes



Finalidade do diagrama de classes

- Modelar os elementos de um programa orientado a objetos em tempo de desenvolvimento
  - Classes com seus atributos e métodos
- Modelar os relacionamentos entre classes, de forma mais explícita que aquela do código
  - Herança
  - Agregação (e composição)
  - Associação

Representação de classe



Representação de atributo

- Exemplos
  - ocupante
  - # ocupante: Jogador
  - # jogadores: Jogador [2..5] {ordenado, único}
- Sintaxe de UML para representação de atributo

  [<**visibilidade** >] [‘/’] <**nome**> [‘:’ < **tipo** >]
  [‘[‘ <**multiplicidade**> ‘]’] [‘=’ <**valor_inicial**>]
  [‘{‘ <**propriedade**> [‘,’ < **propriedade** >]* ’}’]

Representação de método

- Exemplos
  - + iniciar()
  - + alocarPeao (onde : Posicao, quem: Jogador): boolean
- Sintaxe de UML para representação de método

  [<**visibilidade**>] <**nome**> ‘**(**‘
  [<**lista_parametros**>] ‘**)**’
  [‘**:**’ [<**tipo_retorno**>] ‘**{**‘ <**propriedade**> [‘**,**’
  < propriedade >]* ‘**}**’]

Recomendação para estabelecimento de visibilidade

- Atributos → protegidos
  - Princípio da ocultação de informação do paradigma de orientação a objetos
  - Possibilita que atributos herdados ou definidos na classe sejam tratados de maneira uniforme
- Necessidades específicas podem justificar atributos privados
- Atributos públicos → JAMAIS

Recomendação para estabelecimento de visibilidade

- Métodos → públicos
  - O meio externo acessa uma classe através de seus métodos
- Necessidades específicas podem justificar o aumento de restrição de visibilidade

Métodos concretos e abstratos

- **Método concreto** → composto por assinatura e corpo
  - Pode ser invocado em tempo de execução para cumprir a responsabilidade atribuída a ele

Métodos concretos e abstratos

- **Método abstrato** → composto apenas por assinatura
  - Uma declaração de responsabilidade, mas sem a capacidade de cumpri-la, em função da ausência de algoritmo
  - Grafado em itálico

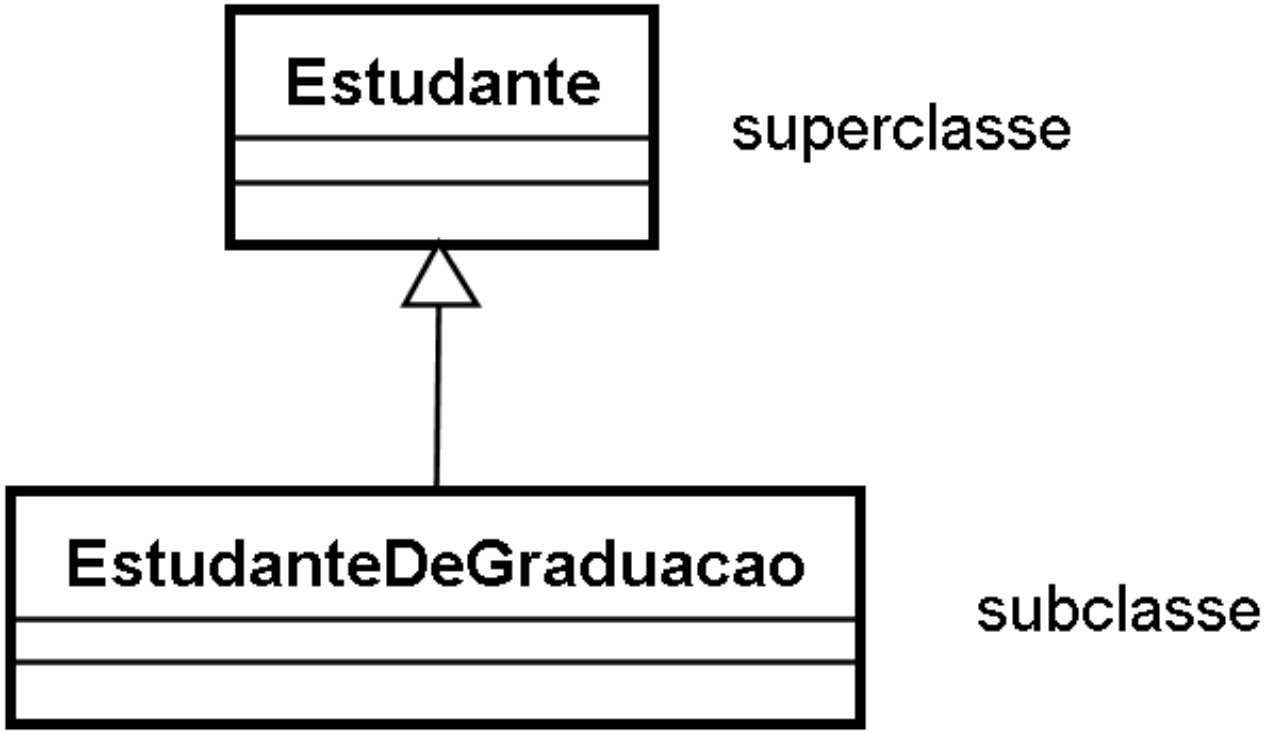
Classes concretas e abstratas

- **Classe concreta** → possui exclusivamente métodos concretos

Classes concretas e abstratas

- **Classe abstrata** → possui pelo menos um método abstrato
  - Identificador da classe grafado em itálico
  - Nem todas as responsabilidades da classe materializadas em capacidades
    - Nem todos os métodos possuem algoritmo definido
    - Não pode originar instâncias em tempo de execução

Relacionamentos – herança



Herança

- Relação de especialização entre DUAS classes
  - Uma delas corresponde a um conceito mais genérico
  - A outra, a um conceito mais específico

Frase característica da herança

<subclasse> é uma espécie de <superclasse>

- No exemplo, estudante de graduação é uma espécie de estudante
  - Estudante de pós-graduação também é uma espécie de estudante
- Se frase não faz sentido, o relacionamento de herança é inadequado
  - Cachorro é uma espécie de gato

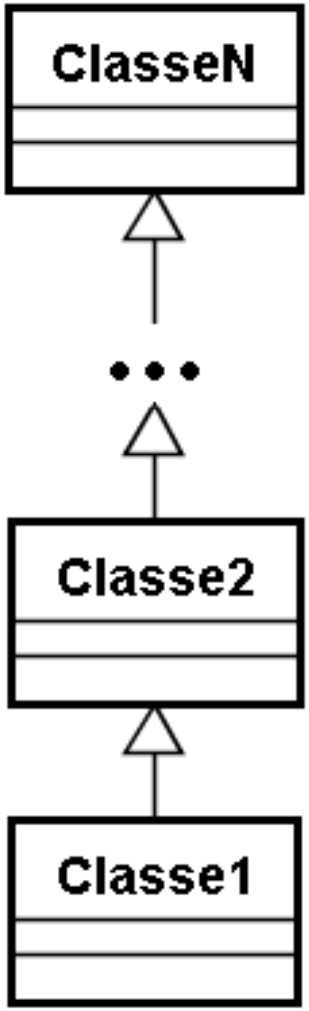
Herança é unidirecional

- Estudante de graduação é uma espécie de estudante
- MAS estudante NÃO é uma espécie de estudante de graduação

Semântica da herança

- Os atributos e métodos da superclasse são **herdados** pela subclasse
  - Os atributos e métodos da superclasse também fazem parte da subclasse
    - Como se tivessem sido definidos nela

Níveis hierárquicos de herança



- Atributos e métodos de ClasseN herdados por todas as classes da hierarquia de herança
- Relacionamento sempre entre 2 classes

Herança de atributos

- É inócuo definir em uma subclasse um atributo com mesmo nome de um atributo de uma superclasse
- Em qualquer nível da hierarquia de herança
- Equivaleria à presença de mais de um atributo com o mesmo nome em uma mesma classe
  - Inconsistente

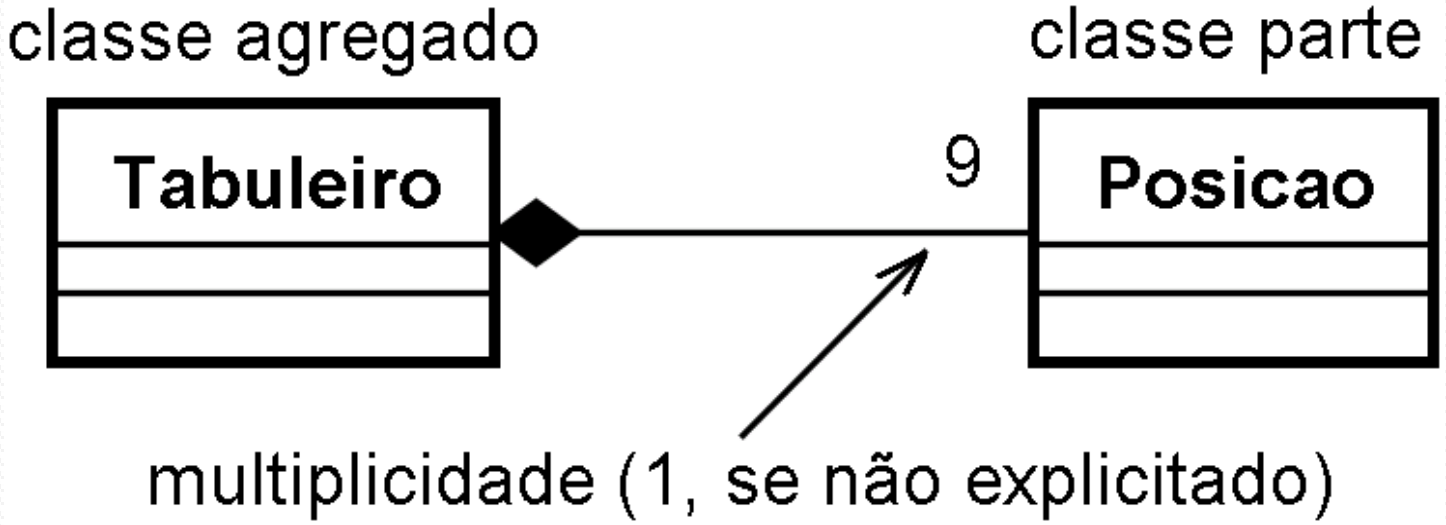
Herança de métodos

- Método em subclasse tem a mesma assinatura de método definido em superclasse → sobrescrição de método
  - Método definido na subclasse substitui o método herdado

Herança e classe abstrata

- **Classe abstrata** → possui pelo menos um método abstrato, que pode ter sido definido na própria classe ou herdado e não sobrescrito
  - O relacionamento de herança precisa ser considerado na definição de classe abstrata

Relacionamentos – agregação e composição



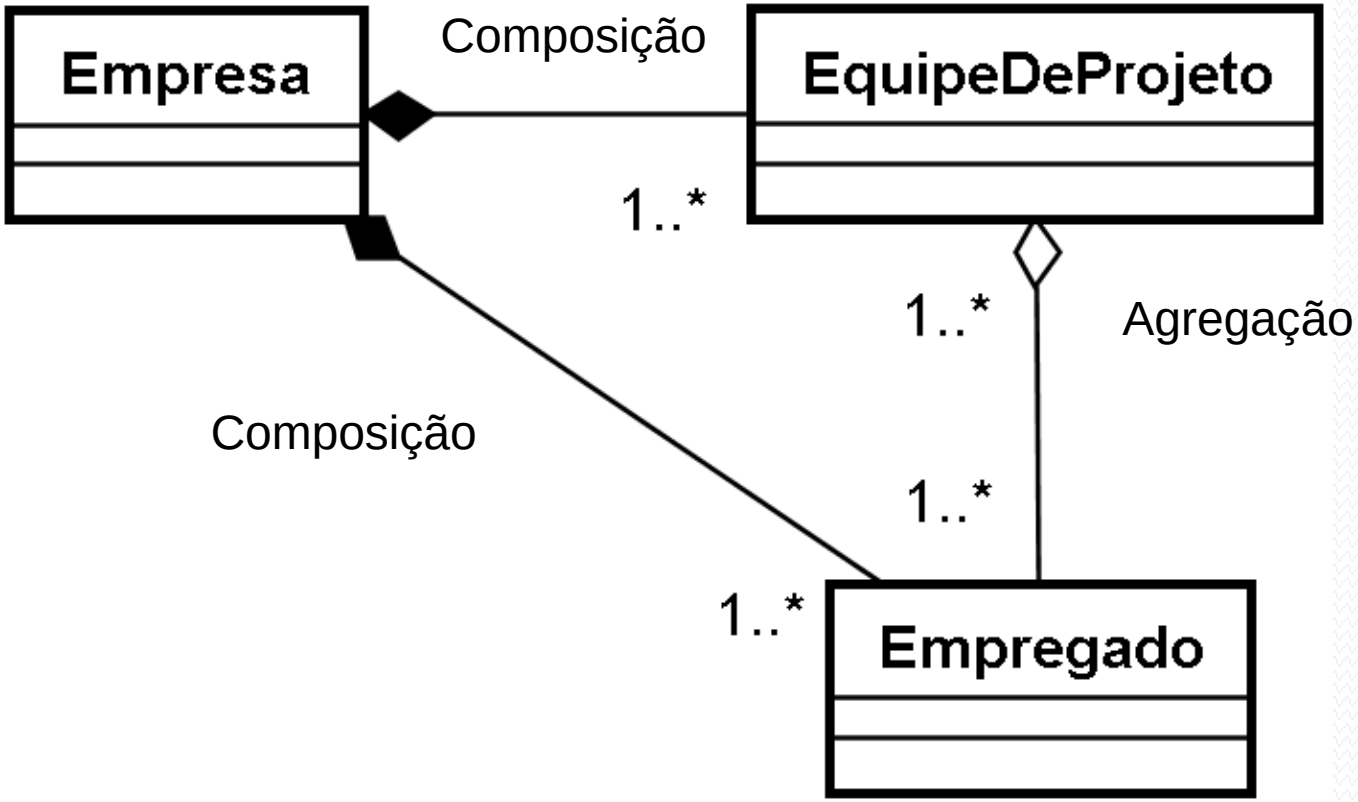
Agregação e composição

- **Agregação** → relacionamento entre DUAS classes que estabelece que uma instância de uma agrupa uma ou mais instâncias da outra
  - Relacionamento todo / parte
-

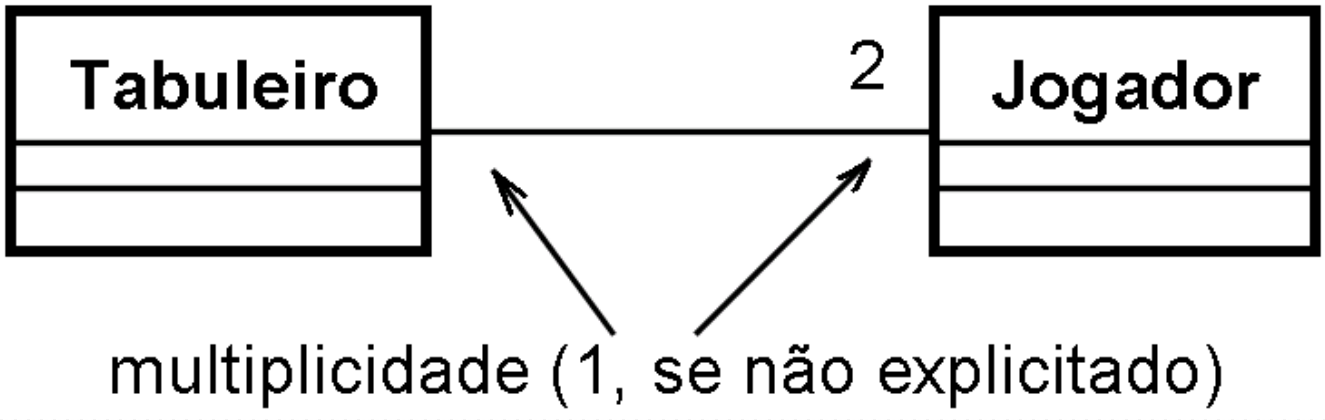
Agregação e composição

- **Composição** → um tipo de relação de agregação com restrições na ligação entre parte e agregado
  - **Uma instância da parte é agregada por uma única instância do agregado**
    - não há compartilhamento da parte
    - Exemplo rodas e carro
  - **A existência da parte depende da existência do agregado**
    - Instanciação do agregado precede a instanciação da parte
    - Destruição do agregado implica na destruição da parte
    - Exemplo tabuleiro e posição

Agregação e composição – exemplo



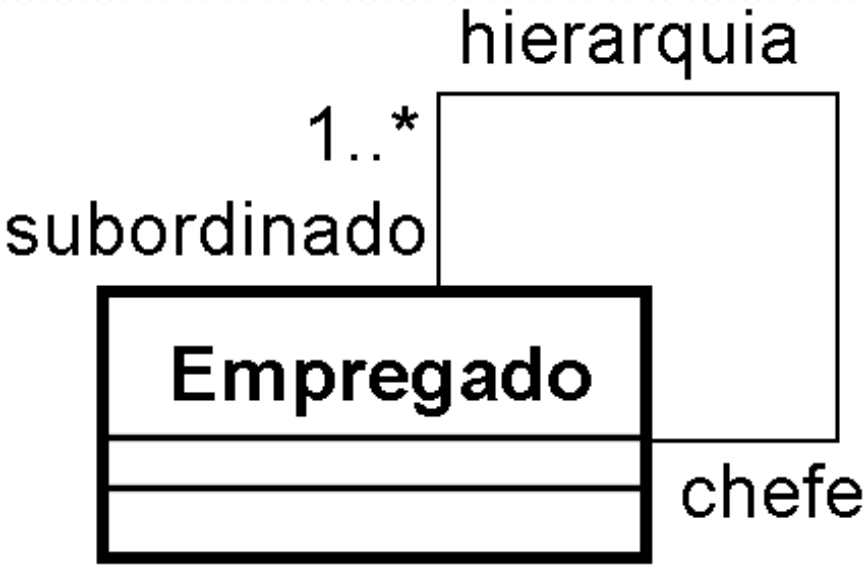
Relacionamentos – associação



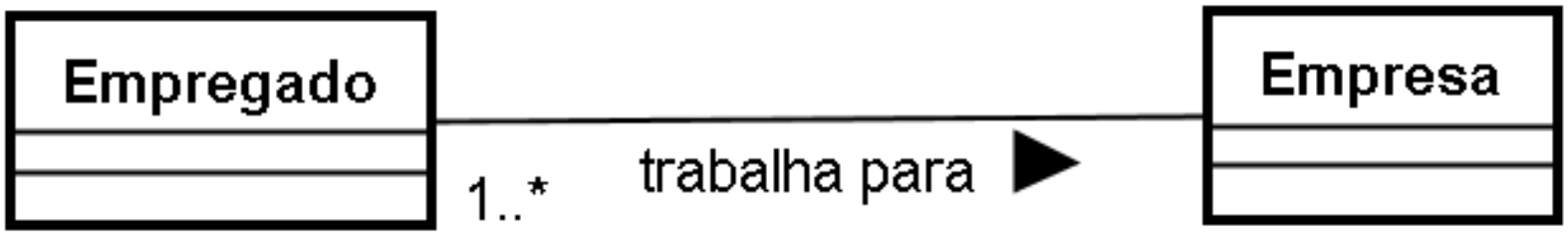
Associação

- Quando há o reconhecimento de um relacionamento entre classes, que não pode ser caracterizado como herança e nem como agregação ou composição
  - Pode ser unidirecional ou bidirecional
  - Pode envolver mais de duas classes
    - A associação entre duas classes é chamada associação binária

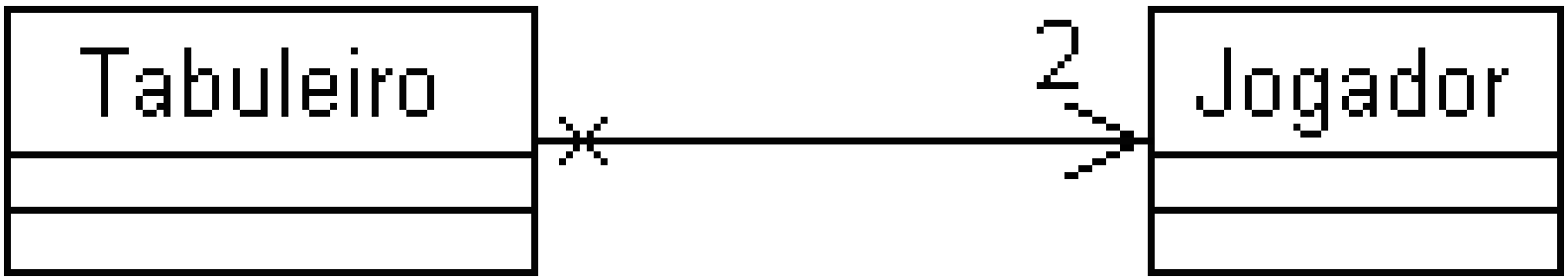
Associação com rótulo e papéis



Associação com indicação de ordem de leitura



Associação com navegabilidade explícita



X : não aponta

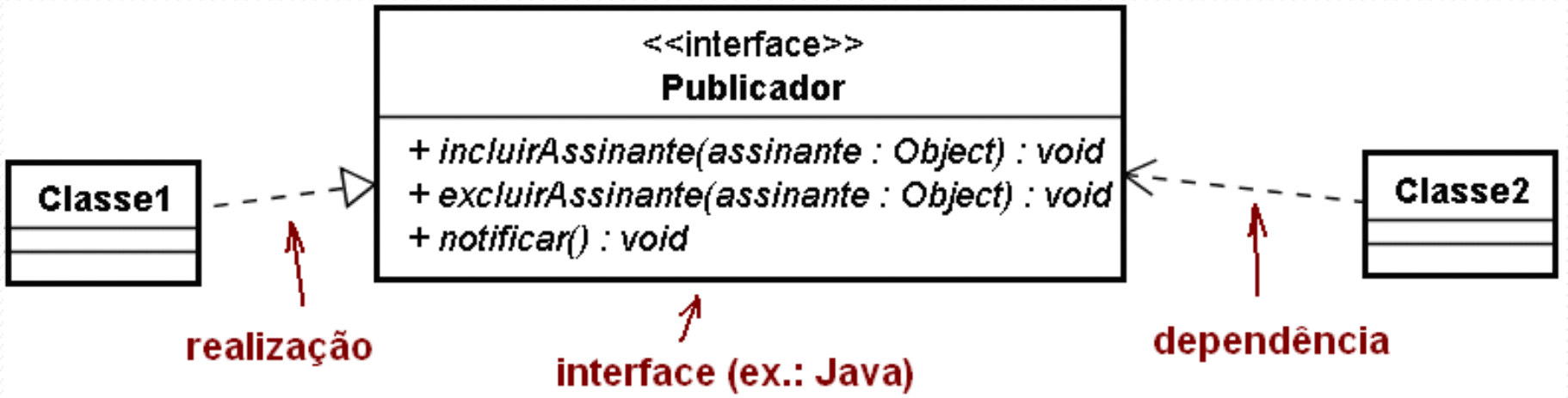
Outros relacionamentos – associações especializadas

- **Realização** → associação entre dois elementos em que um deles especifica uma responsabilidade a ser implementada e o outro incorpora a obrigação de implementá-la
  - Usado para associar uma classe a uma interface

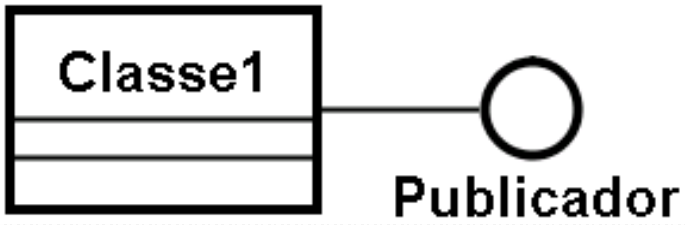
Outros relacionamentos – associações especializadas

- **Dependência** → associação entre dois elementos em que um deles é declarado como dependente daquilo que deve estar implementado em outro
  - Um é cliente dos serviços oferecidos pelo outro
  - Usado para relacionar outros elementos além de classes (como interface e classe)
  - Usado também em outros diagramas

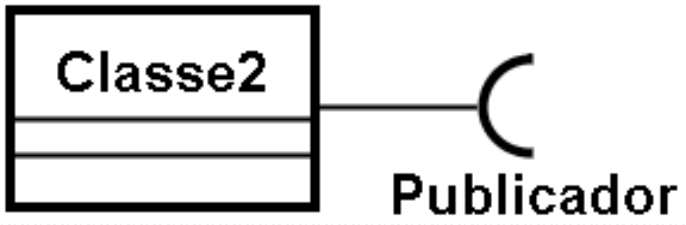
Interfaces (semântica de Java)



Vinculação de classes a uma interface previamente definida



realização



dependência

Considerações sobre esta aula

- Diagrama de classes → modelagem gráfica mais primitiva de um programa orientado a objetos
  - Reflete a estrutura do código
- Elementos sintáticos do diagrama
  - Classes
  - Relacionamentos
- Conhecer os elementos do diagrama é requisito para usá-lo em modelagem

Referências

Booch, G.; Jacobson, I. e Rumbauch, J. **UML: Guia do Usuário**. Campus, 2006.

Silva, R. P. **UML 2 em modelagem orientada a objetos**. Visual Books, 2007.