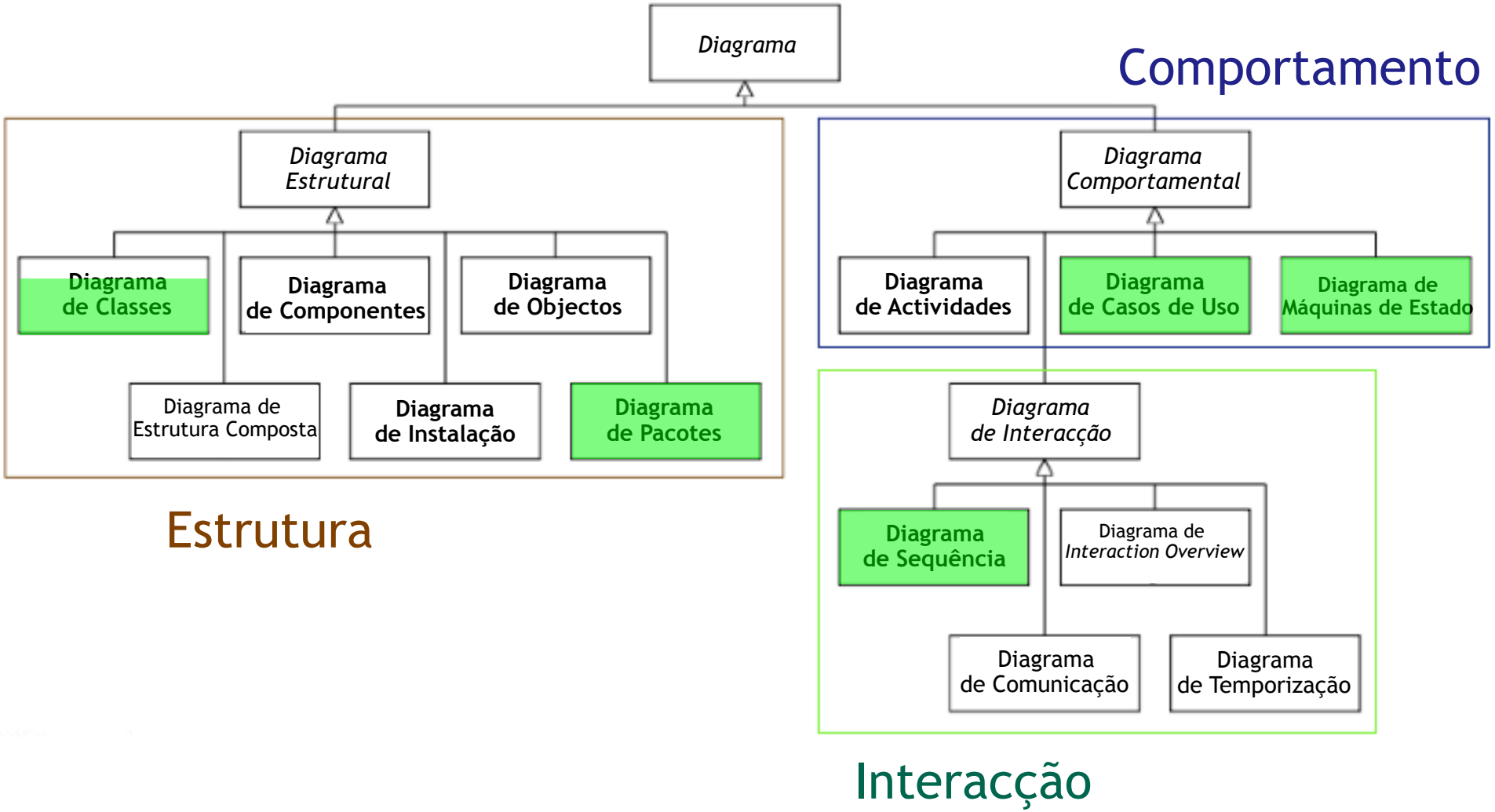
Desenvolvimento de Sistemas Software

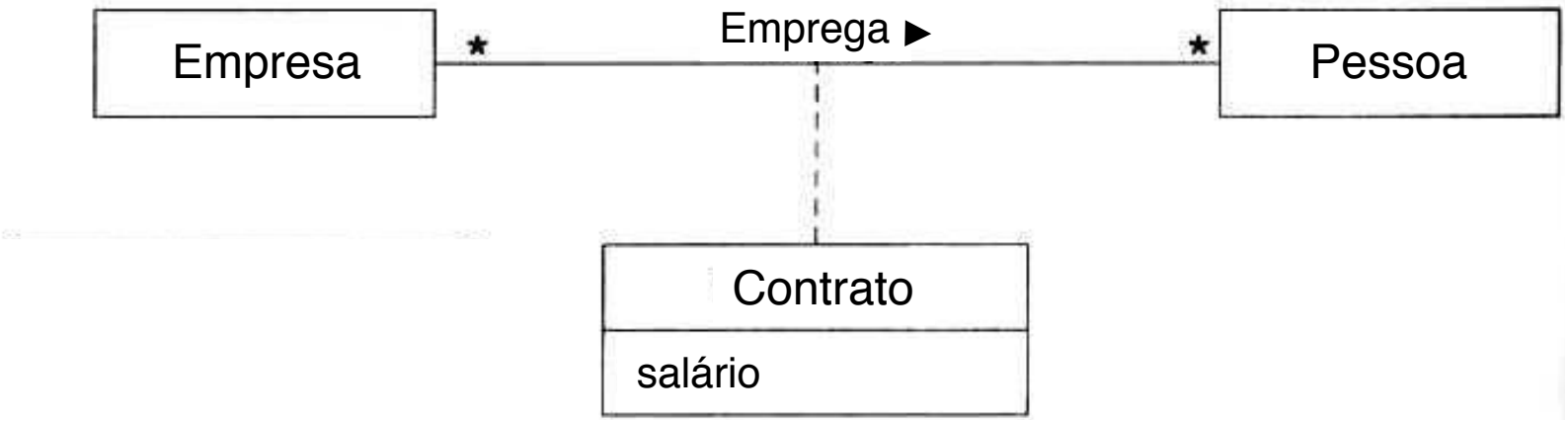
Aula Teórica 18: Modelação Estrutural / Diagramas de Classe III

Desenvolvimento de Sistemas Software
José Creissac Campos/António Nestor Ribeiro

Diagramas da UML 2.x



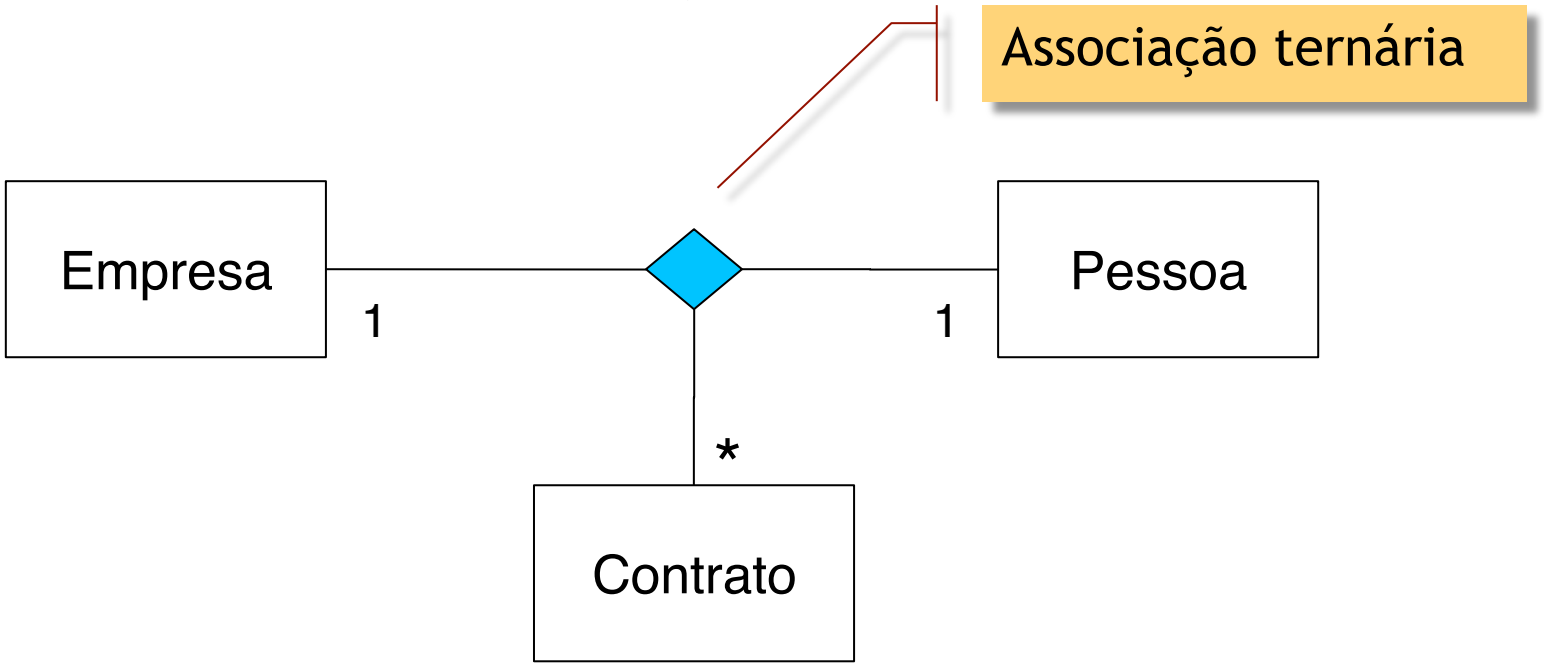
Da aula anterior… Classes de Associação



- Cada Pessoa pode ter estado contratada por várias Empresas e para cada uma há um contrato diferente.
- O Contrato não é característica da Empresa, nem da Pessoa, mas da relação entre ambas.
- **Dois contratos diferentes com a mesma empresa?!**

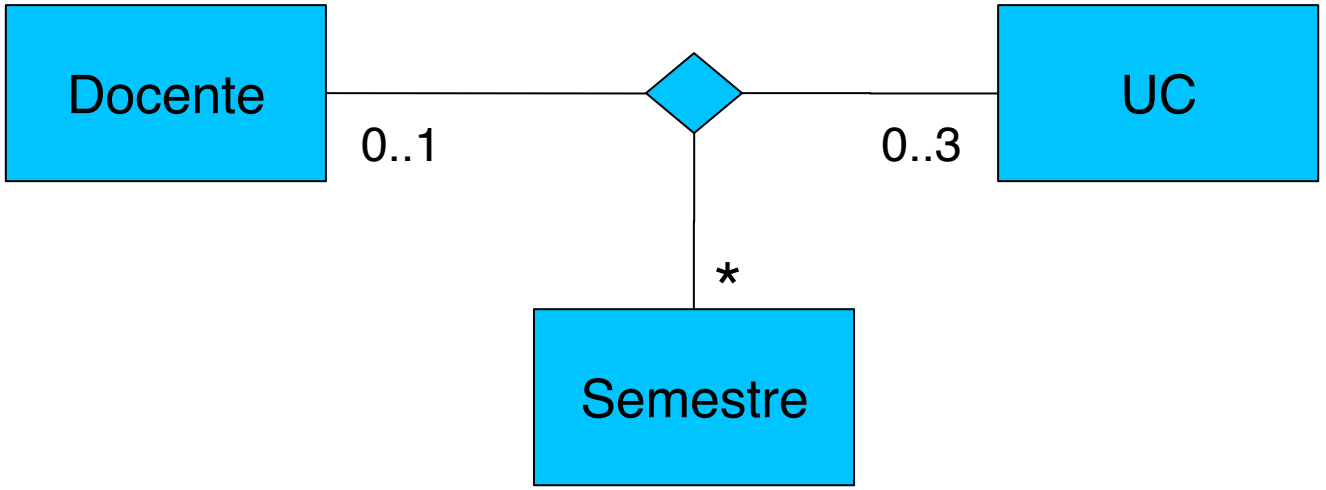
Associações n-árias

- A UML não se restringe a associações binárias:



- Multiplicidades indicam quantos objectos existem para uma dada combinação de objectos das outras classes.
- Navegabilidade, agregação e qualificação **não são** permitidos.

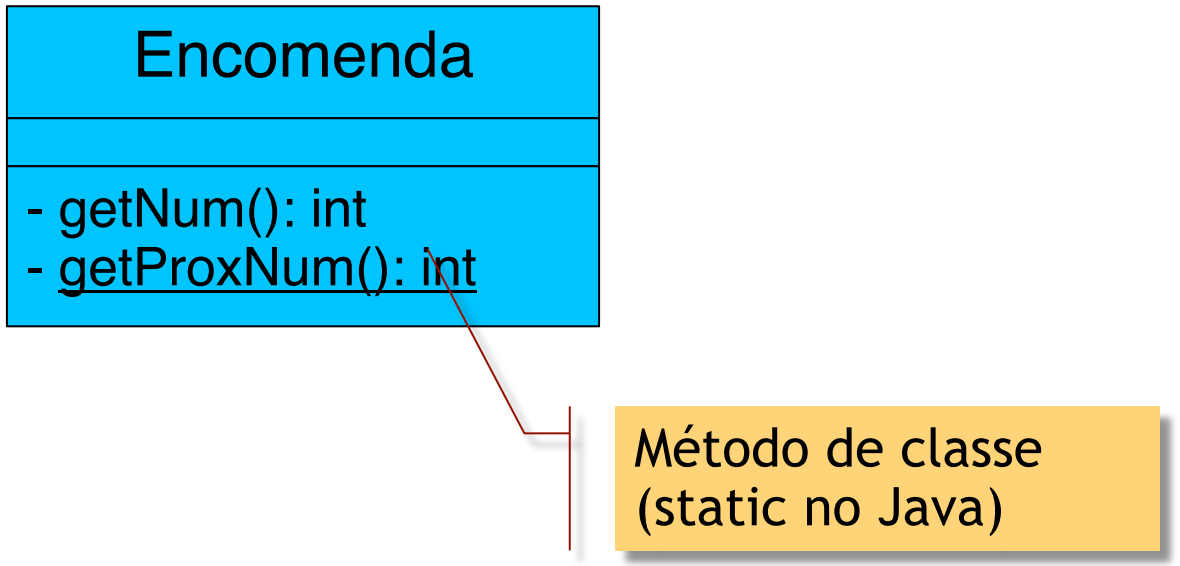
Associações n-árias - outro exemplo



- Cada docente pode leccionar num semestre, no máximo, três UCs.
- Uma multiplicidade de zero invalida a combinação de objectos(!)
  - Não é possível ter uma associação entre uma UC e um Semestre sem indicar o Docente
  - Não é possível dizer que um Docente dá aulas num Semestre sem indicar , pelo menos, uma UC

Operações e variáveis de classe

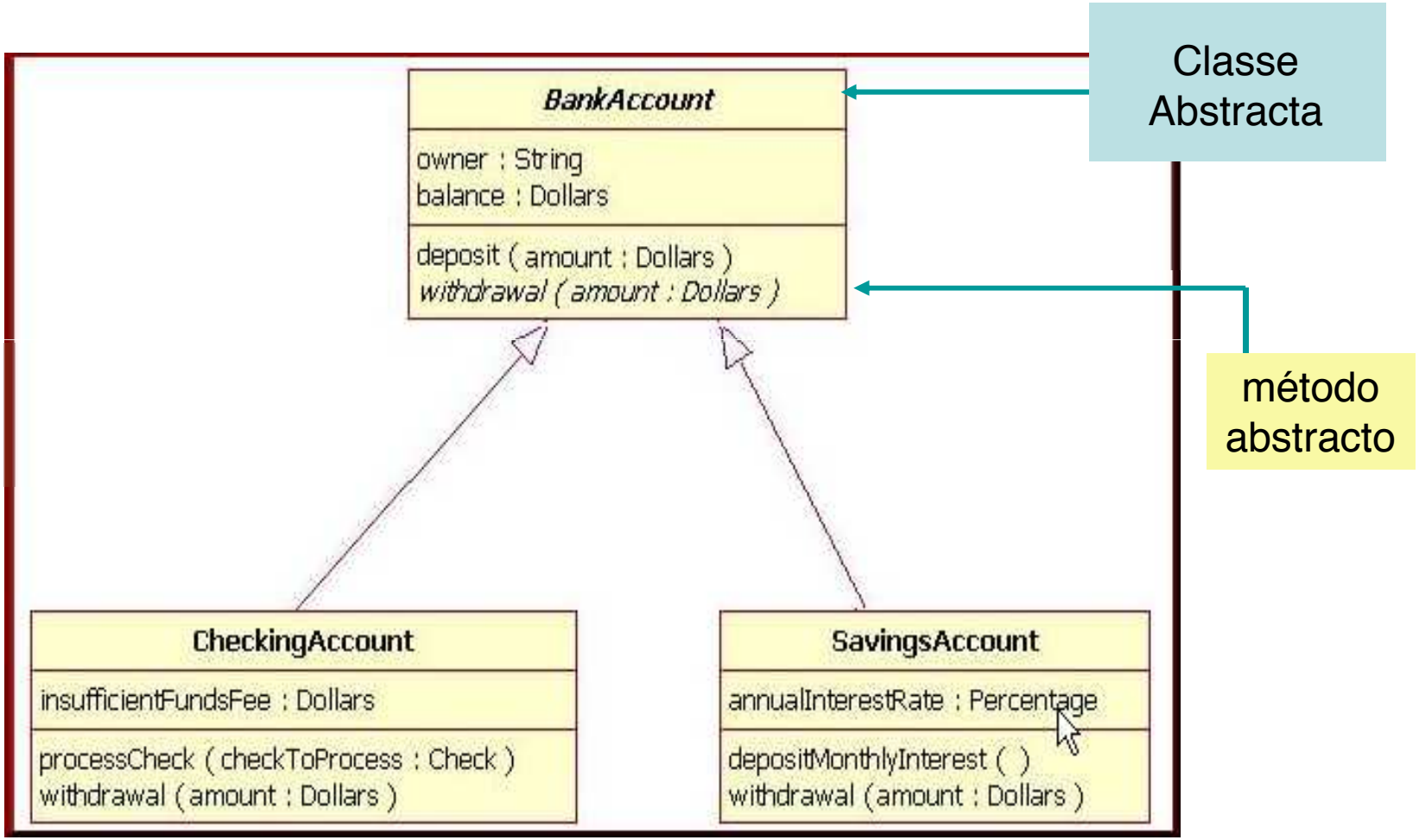
- Variáveis de classe são variáveis globais a todas as instâncias de uma classe.
- Métodos de classe são métodos executados directamente pela classe e não por uma das suas instâncias (logo, não têm acesso directo a variáveis/métodos de instância).
- São representados tal como variáveis/métodos de instância, mas sublinhados.
- Deve evitar-se abusar de operações e variaveis de classe.



Classes abstractas

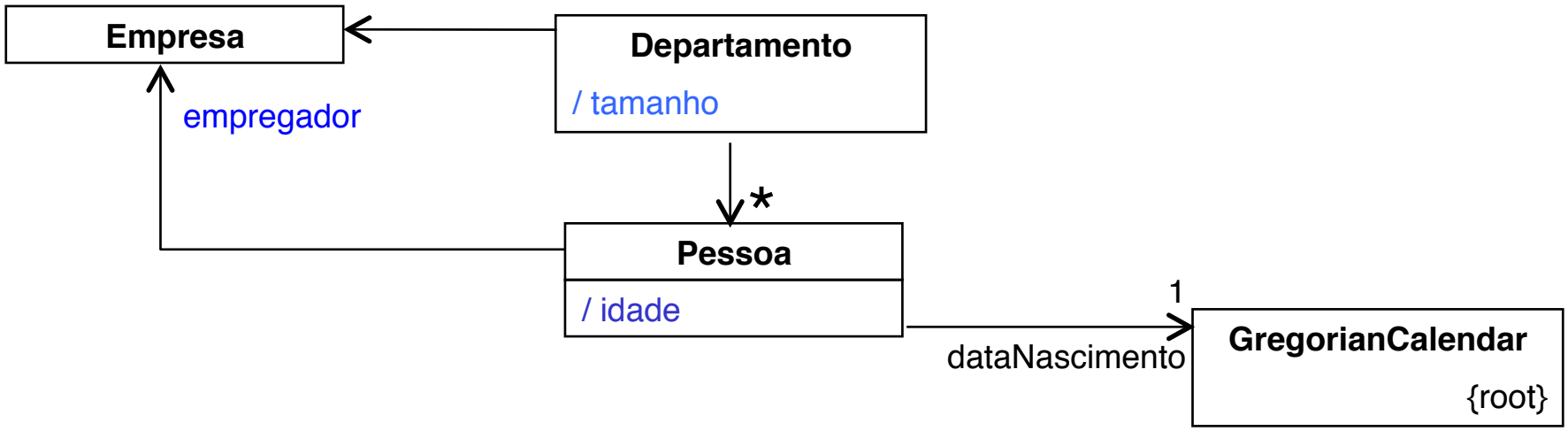
- Nem sempre ao nível da super-classe é possível saber qual deverá ser o método associado a uma operação.
- Quando se está a utilizar uma hierarquia de classes para representar sub-tipos, pode não fazer sentido permitir instâncias da super classe.
- Uma operação abstracta é uma operação que não tem método associado na classe em que está declarada.
- Uma classe abstracta é uma classe da qual não se podem criar instâncias e que pode conter operações abstractas.
- Classes concretas (não abstractas) não podem conter métodos abstractos!
- Notação: em *itálico* ou através da propriedade {abstract}.

Exemplo


BankAccount
owner : String
balance : Dollars
deposit (amount : Dollars)
withdrawal (amount : Dollars)
CheckingAccount
insufficientFundsFee : Dollars
processCheck (checkToProcess : Check)
withdrawal (amount : Dollars)
SavingsAccount
annualInterestRate : Percentage
depositMonthlyInterest ( )
withdrawal (amount : Dollars)
Classe Abstracta
método abstracto

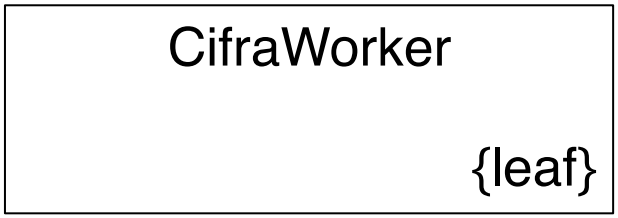
Classes root

- Classes etiquetadas com a propriedade {root} não podem ser generalizadas.
  - Por exemplo, se o modelo apresenta classes pertencentes ao ambiente de desenvolvimento que irá ser utilizado, não será viável generalizar tais classes.



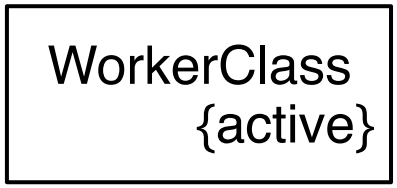
Classes leaf

- Classes etiquetadas com a propriedade {leaf} não podem ser especializadas (classes final no Java).
  - Por exemplo, se o sistema contém uma classe que fornece serviços de encriptação, por motivos de segurança não é desejável que os métodos associados às operações dessa classe possam ser redefinidos (isto também pode ser controlado ao nível das operações).



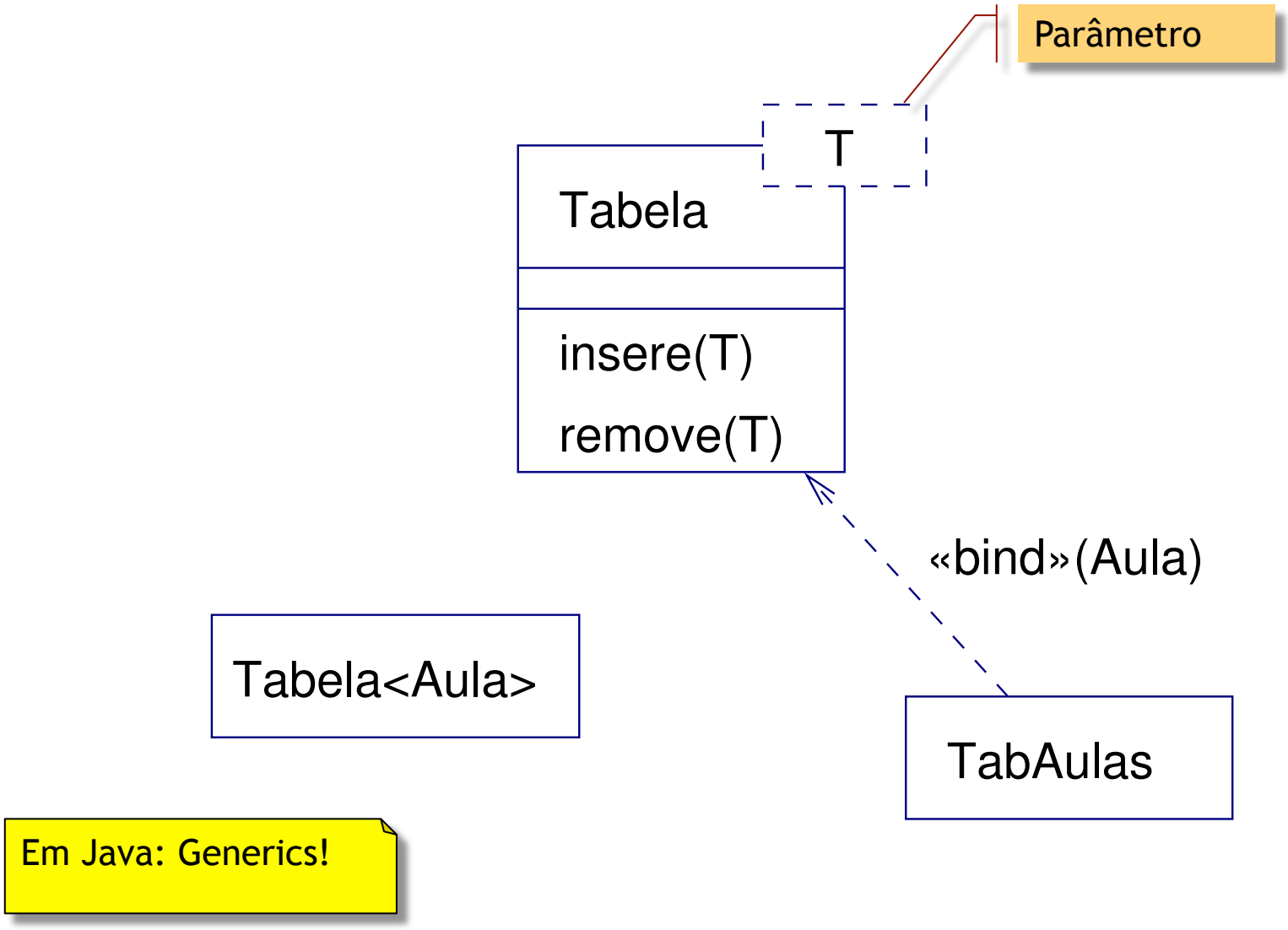
Classes active

- Classes etiquetadas com a propriedade {active} são consideradas ativas
  - Por exemplo, uma *thread*.



- *Notação gráfica*

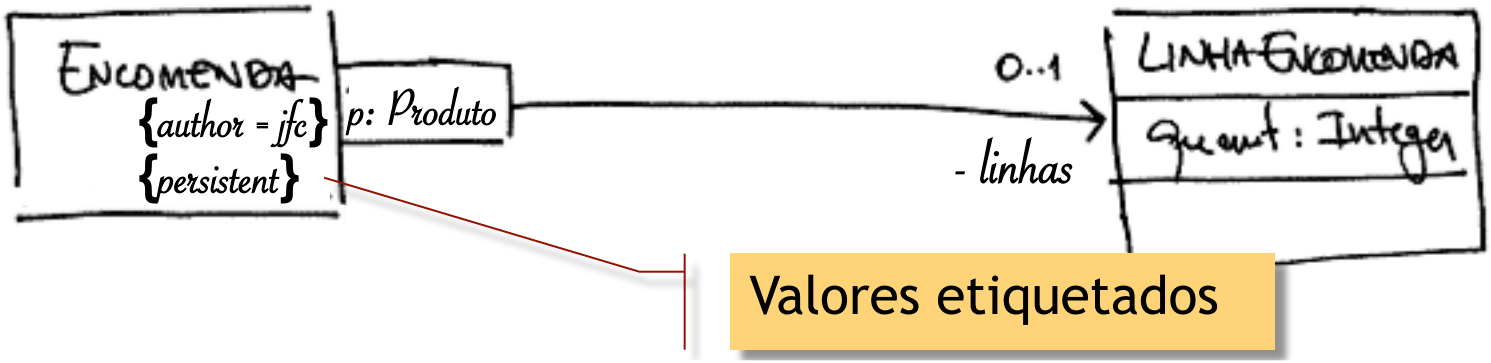
Classes parametrizadas (*Template classes*)



Mecanismos de extensibilidade

- "Tagged Values" (valores etiquetados)
- Estereótipos
- Restrições ("constraints")

- Valores Etiquetados
  - Definem novas propriedades das "coisas"
  - Trabalham ao nível dos meta-dados

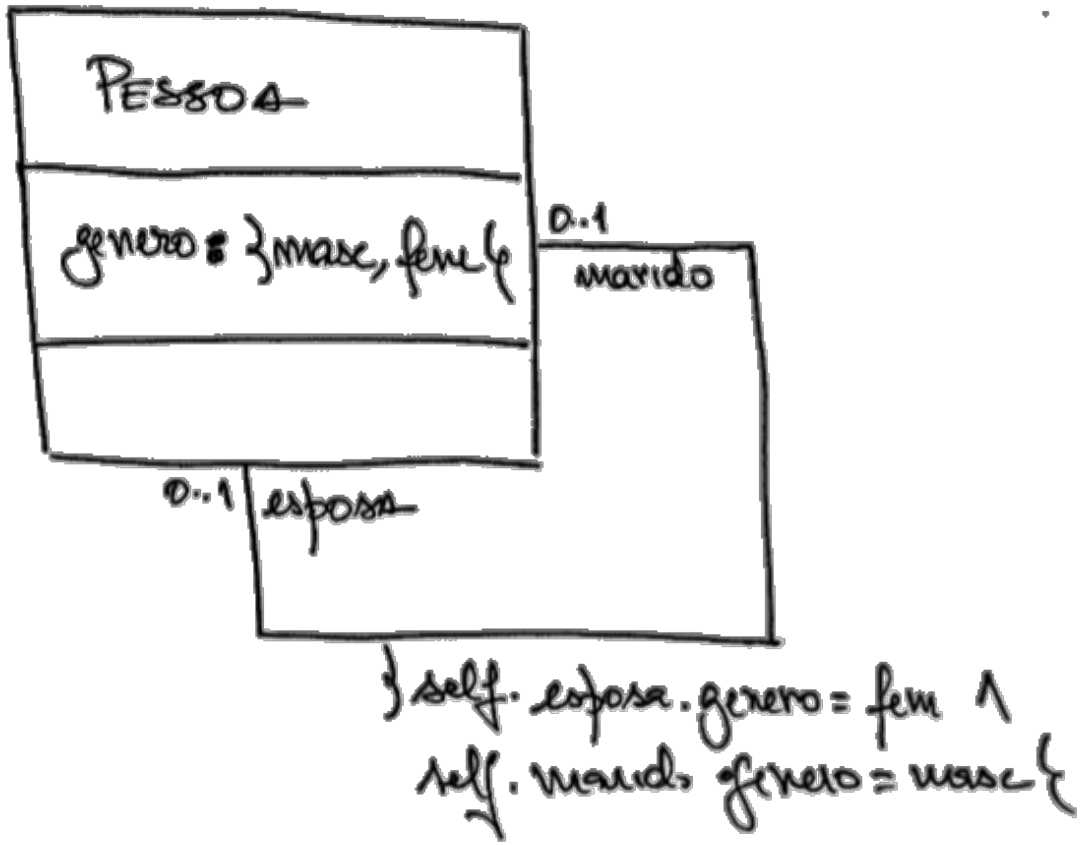


Mecanismos de extensibilidade

- **Estereótipos**
  - Permitem a definição de variações dos elementos de modelação existentes (ex: «include», «extend» são estereótipos de dependência)
  - Possibilitam a extensão da linguagem de forma controlada
  - Cada estereótipo pode ter a si associado um conjunto de valores etiquetados
    - Trabalham ao nível dos meta-dados
  - Meta-tipo de dados ≠ Generalização

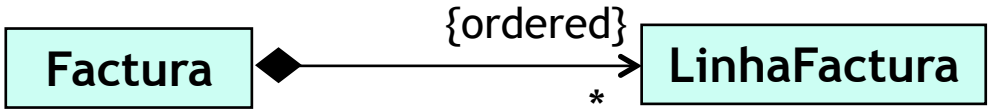
Mecanismos de extensibilidade

- **Restrições**
  - Utiliza-se quando a semântica das construções diagramáticas do UML não é suficiente



Restrições às associações

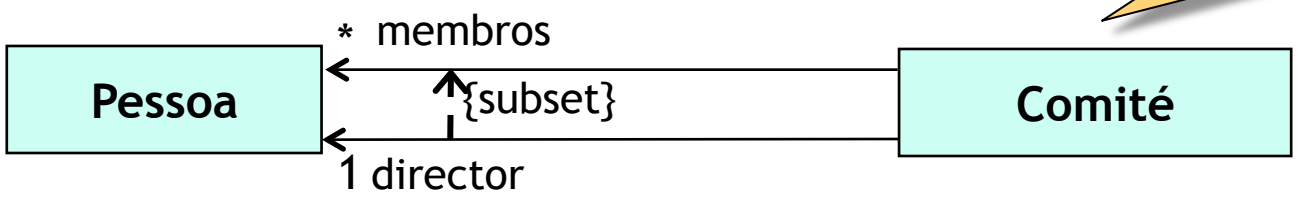
uma factura é constituída por um conjunto ordenado de 0 ou mais linhas



Restrição aplicada a um dos papeis (*roles*).

E.g. ordered, sorted

o director de um comité tem que ser um dos seus membros

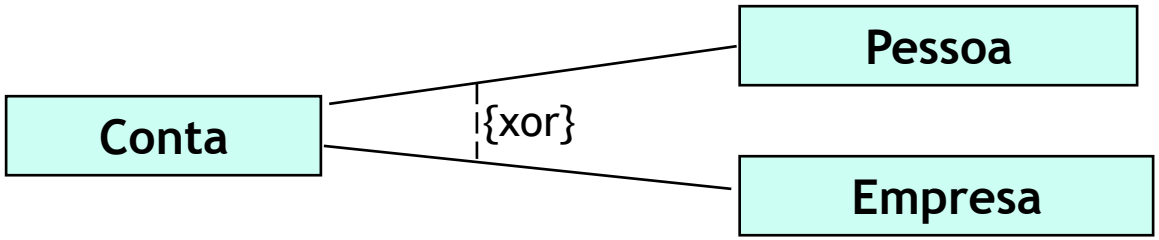


Restrição aplicada a duas associações (com direcção).

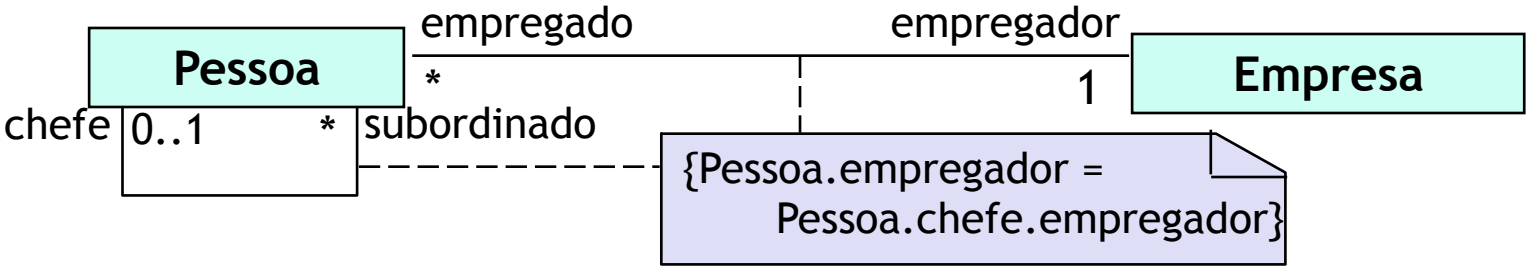
Restrições às associações

Restrições aplicadas a duas associações (sem direção). E.g. associações mutuamente exclusivas.

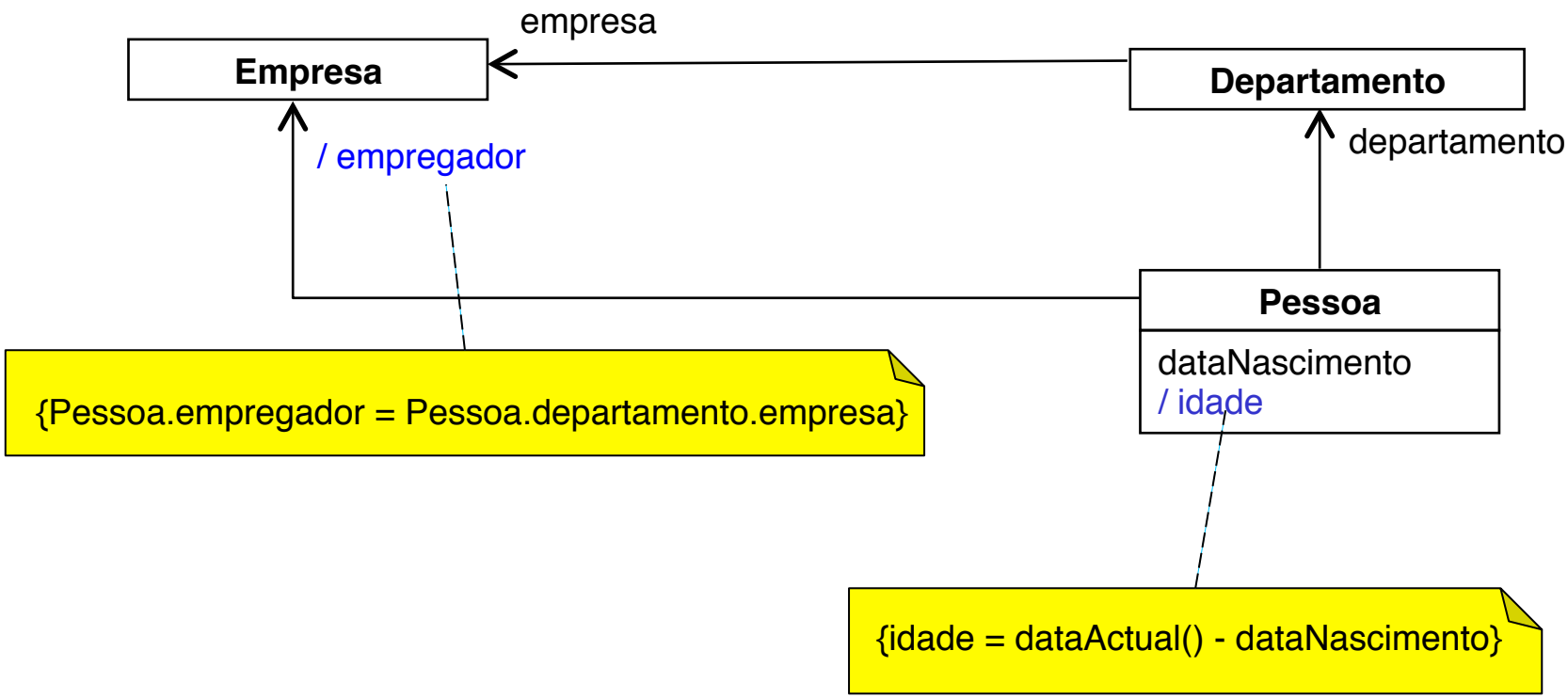
uma conta pode ser de uma pessoa ou de uma empresa (mas não de ambos)



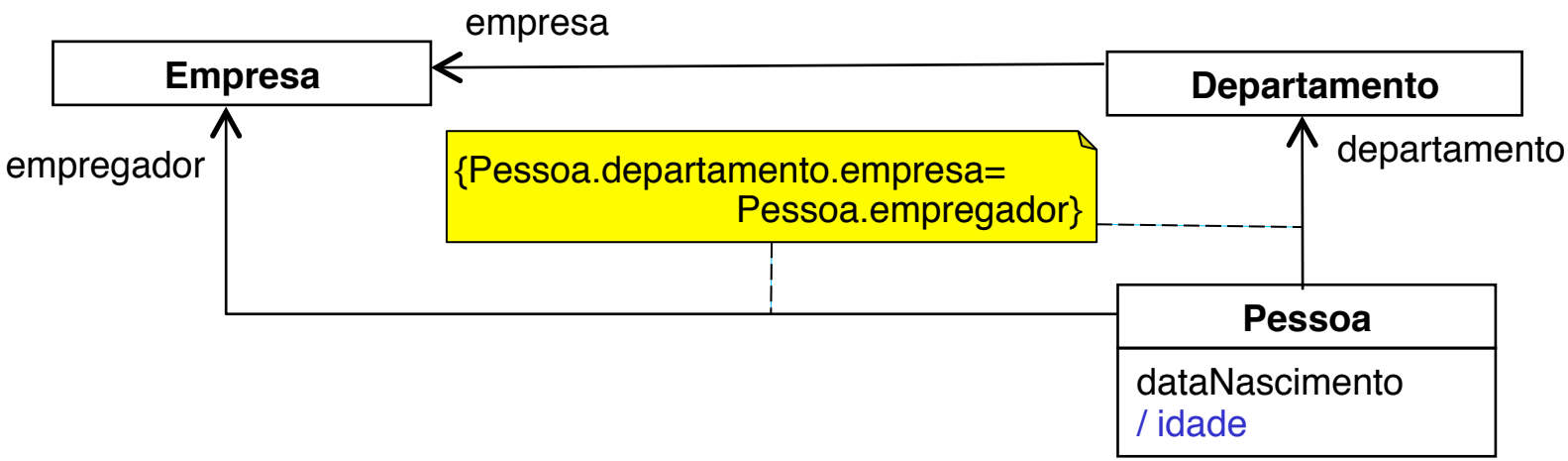
o empregador do chefe, é o empregador do subordinado



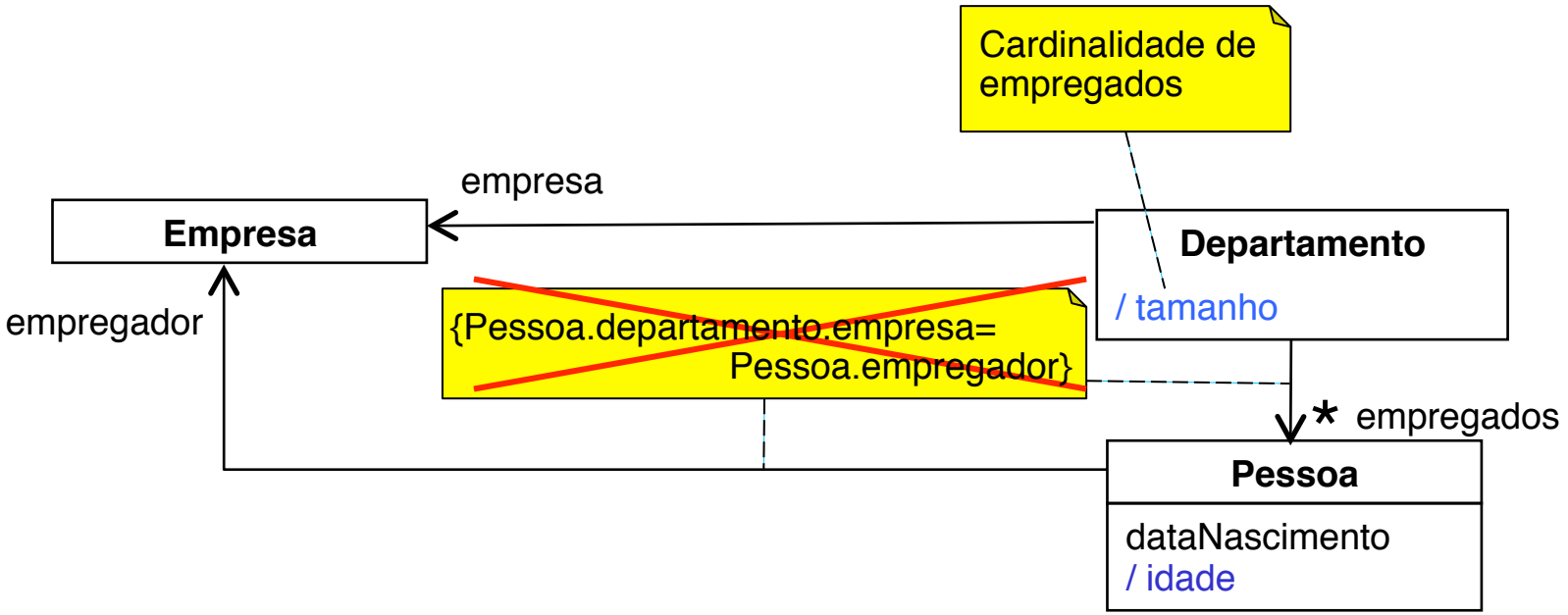
Exemplos de restrições



Exemplo restrições



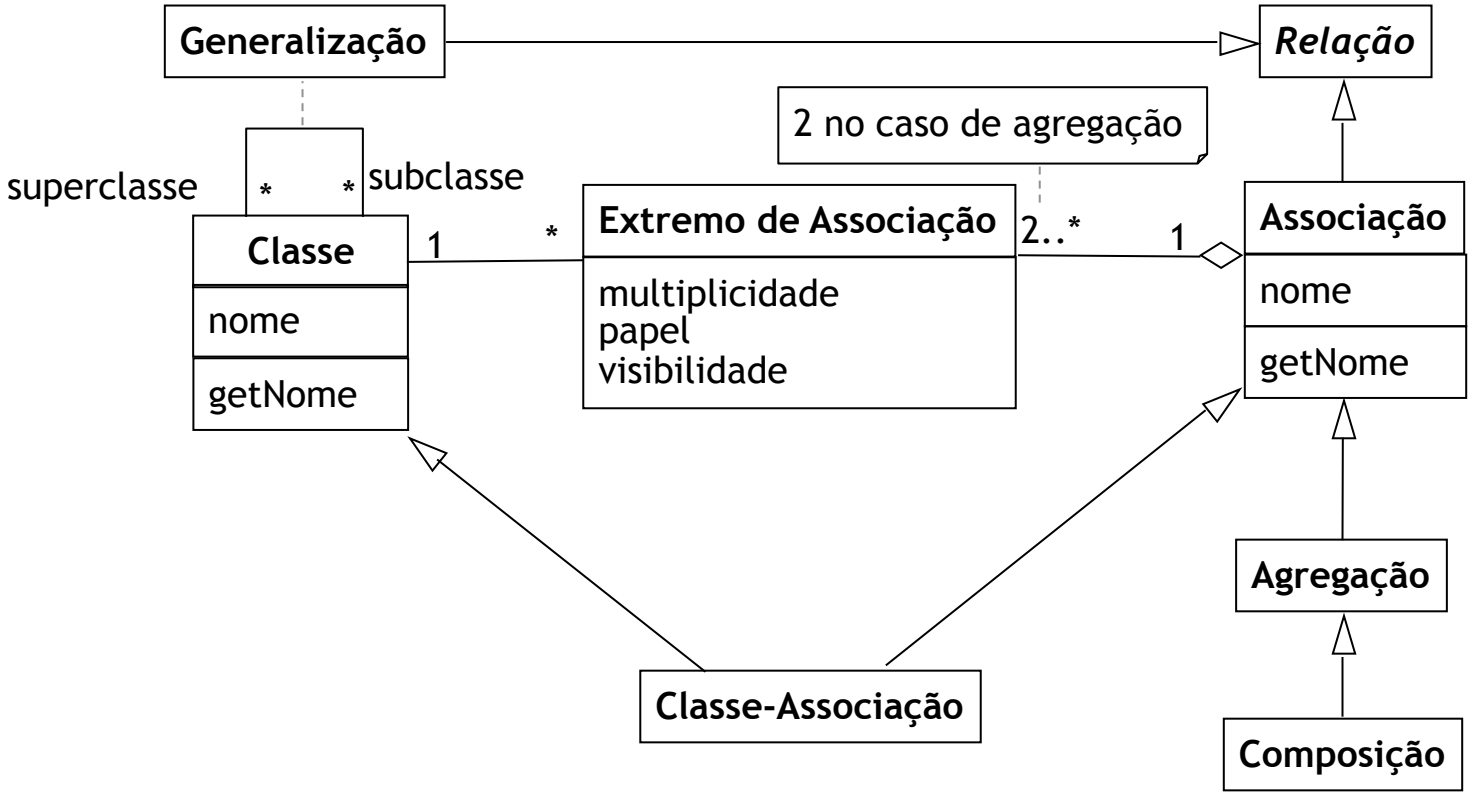
Exemplos de atributos derivados



- Restrições tem que respeitar navegabilidade
  - Alteração na navegabilidade impossibilita esta restrição
  - Onde colocar restrição e qual?

Meta-modelo do UML (relações entre classes)

- É possível descrever em UML a semântica dos diagramas



Interfaces

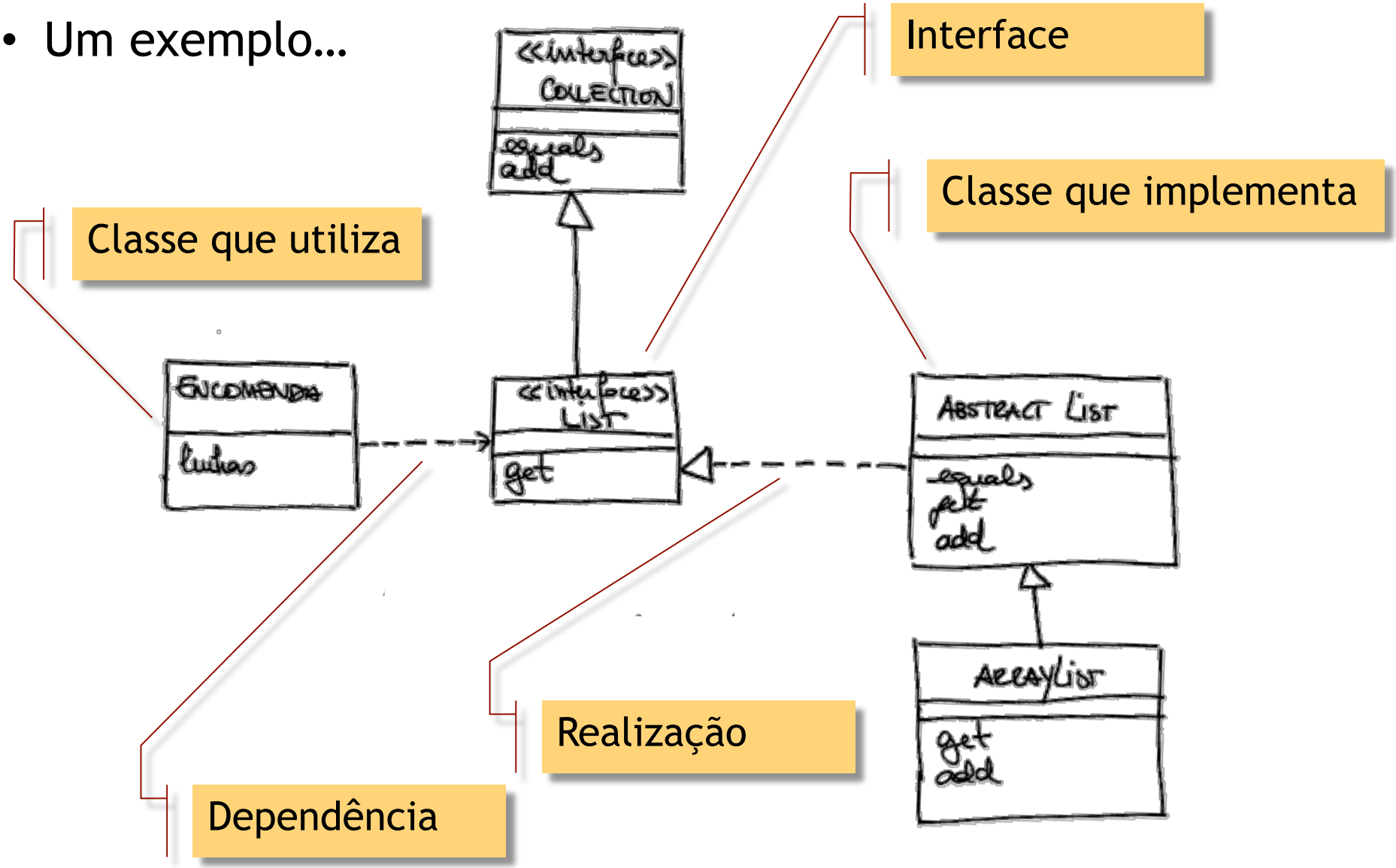
- Uma interface especifica um tipo abstracto – um conjunto de operações externamente visíveis que uma classe (ou componente, subsistema, etc.) deve implementar
- semelhante a classe abstracta só com operações abstractas e sem atributos nem associações
- separação mais explícita entre interface e (classes de) implementação
- interfaces são mais importantes em linguagens que têm herança simples de implementação e herança múltipla de interface (como em Java)

Interfaces

- Relação de concretização de muitos para muitos entre interfaces e classes de implementação
- Vantagem em separar interface de implementação: os clientes de uma classe podem ficar a depender apenas da interface em vez da classe de implementação
- Notação UML:
  - classe com estereótipo «interface» (ligada por relação de realização à classe de implementação), ou
  - notação “lollipop” – círculo (ligado por linha simples à classe de implementação).

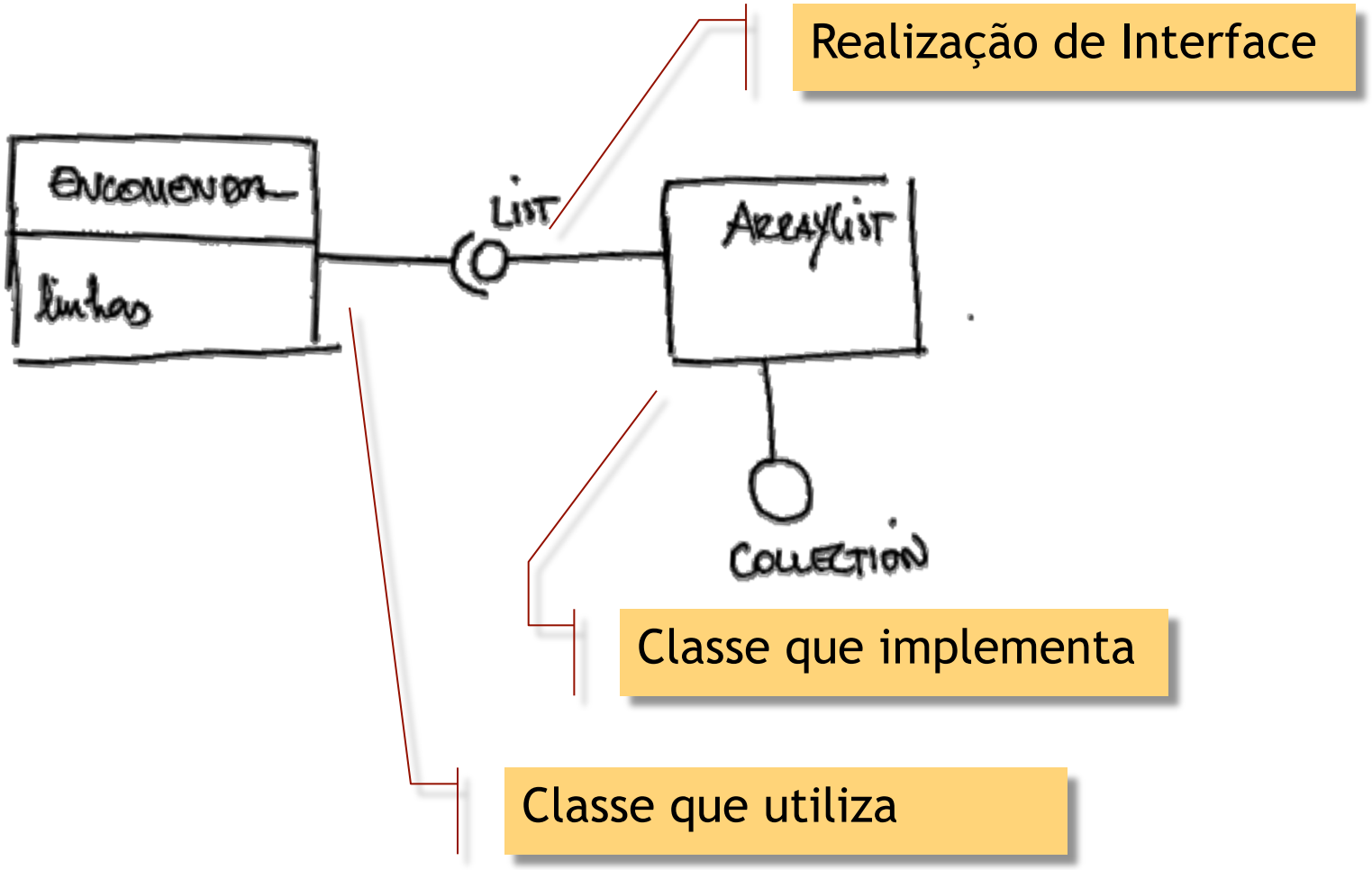
Interfaces

- Um exemplo…

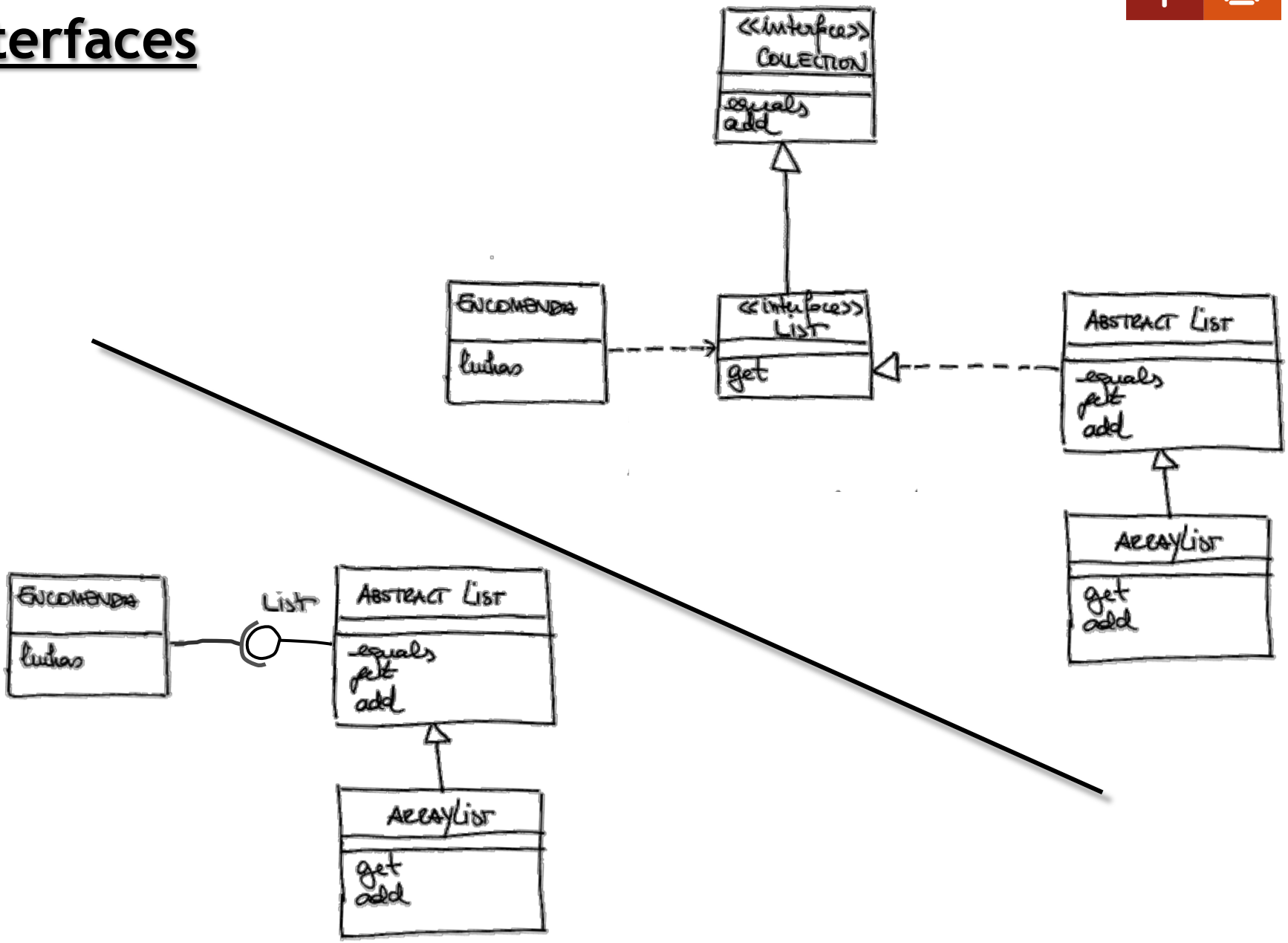


Interfaces

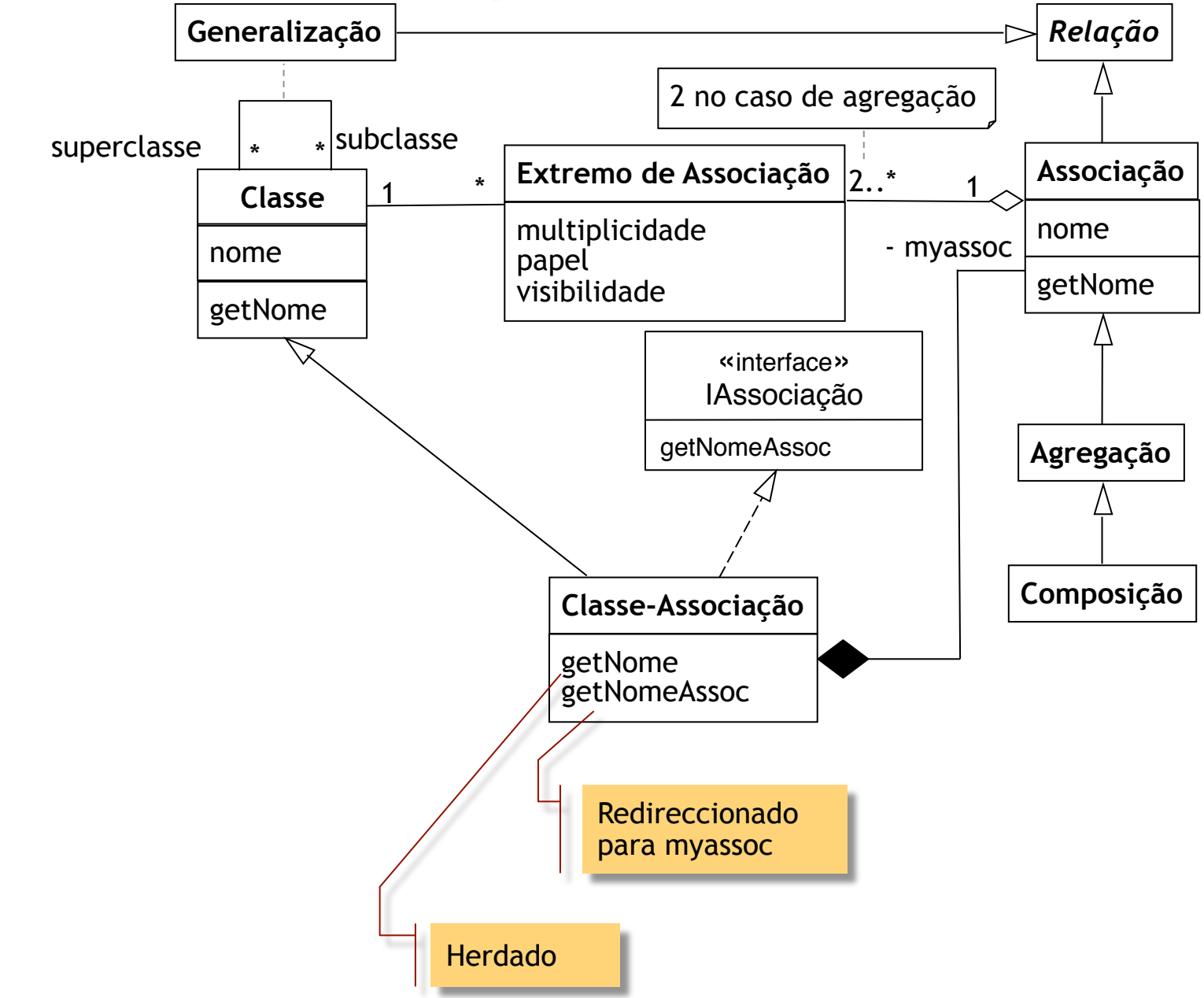
- Notação “lollipop” do UML 2.x



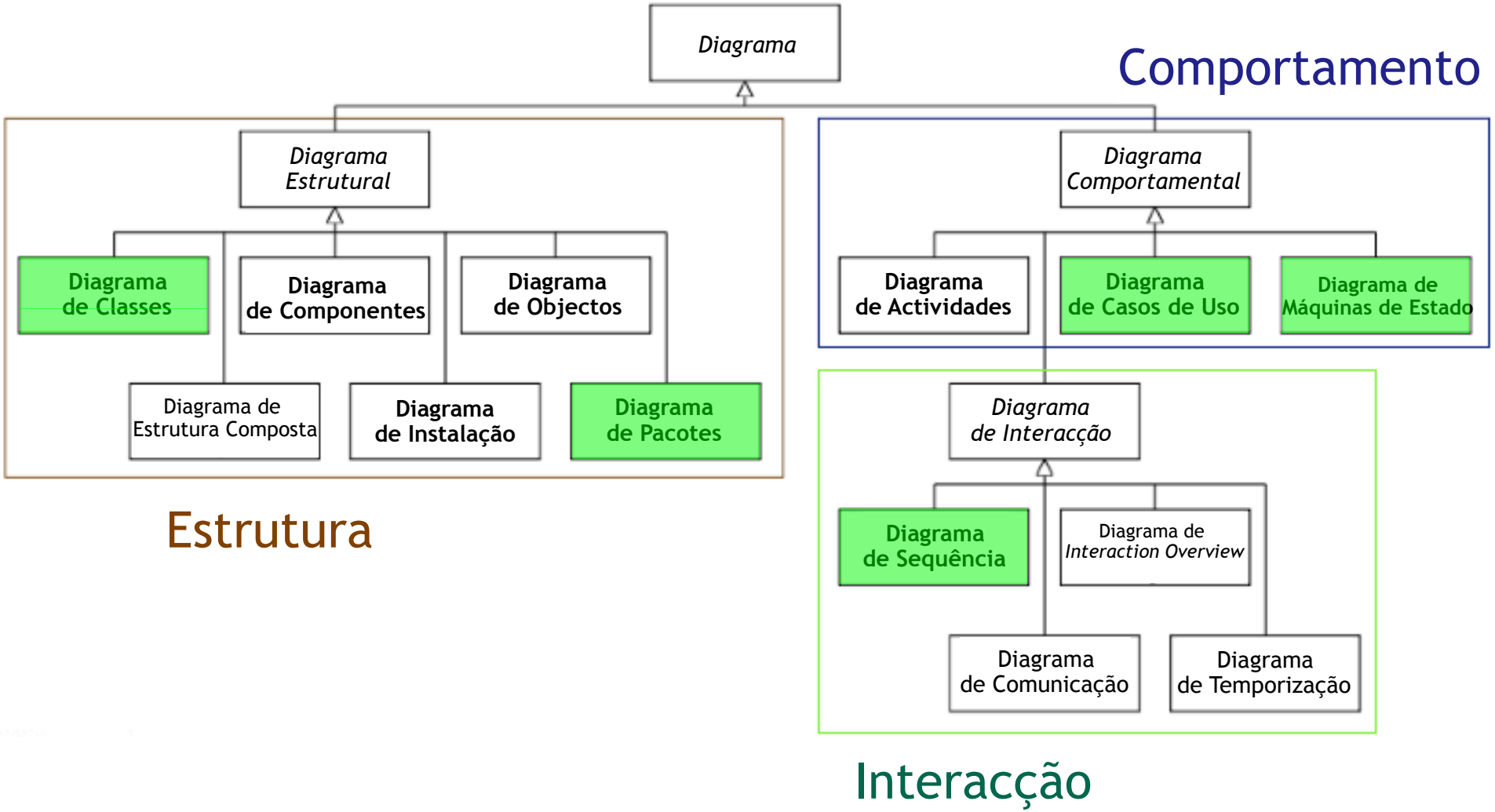
Interfaces


«interface»
COLLECTION
equals
add
ENCOMENDAS
linhas
«interface»
LIST
get
ABSTRACT LIST
equals
get
add
ARRAYLIST
get
add
ENCOMENDAS
linhas
List
ABSTRACT LIST
equals
get
add
ARRAYLIST
get
add

Interfaces - exemplo



Diagramas da UML 2.x



Modelação Estrutural

Sumário

- Diagramas de Classe III
  - Associações n-árias
  - Operações e variáveis de classe
  - Classes abstratas
  - Classes root, leaf e active
  - Classes parametrizadas
  - Mecanismos de extensibilidade: valores etiquetados, estereótipos e restrições
  - Restrições às associações
  - Interfaces