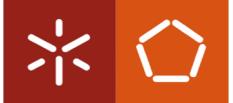




Quiz 1

- 61 respostas
- Pontos possíveis: 40
 - Pontuação media: 12!
- Um Modelo de Use Case...
 - consiste num Diagrama de Use Case - **8%**
 - **permite pensar de que forma o sistema irá suportar as actividades dos utilizadores** - 61%
 - **captura requisitos funcionais** - 50%
 - captura requisitos não-funcionais - **2%**
- A qualidade de um Diagrama de Use Case está positivamente correlacionada com a quantidade de «include» e «extend» que existem no diagrama
 - Verdadeiro - 7%
 - **Falso - 80%**
- A UML...
 - é um processo de desenvolvimento - 36%
 - **é definida numa norma OMG - 28% !!**
 - permite, com uma só linguagem gráfica, modelar diferentes aspectos de um Sistema - **67% !!**
 - foi pensada especificamente para a programação em Java - **0%**
- Um Modelo de Domínio...
 - é um diagrama UML - **51% !!**
 - permite analisar as funcionalidade de um sistema - **10%**
 - **pode ser descrito com um Diagrama de Classes** - **41%**
 - **caracteriza as entidades de um dado problema** - **48%**

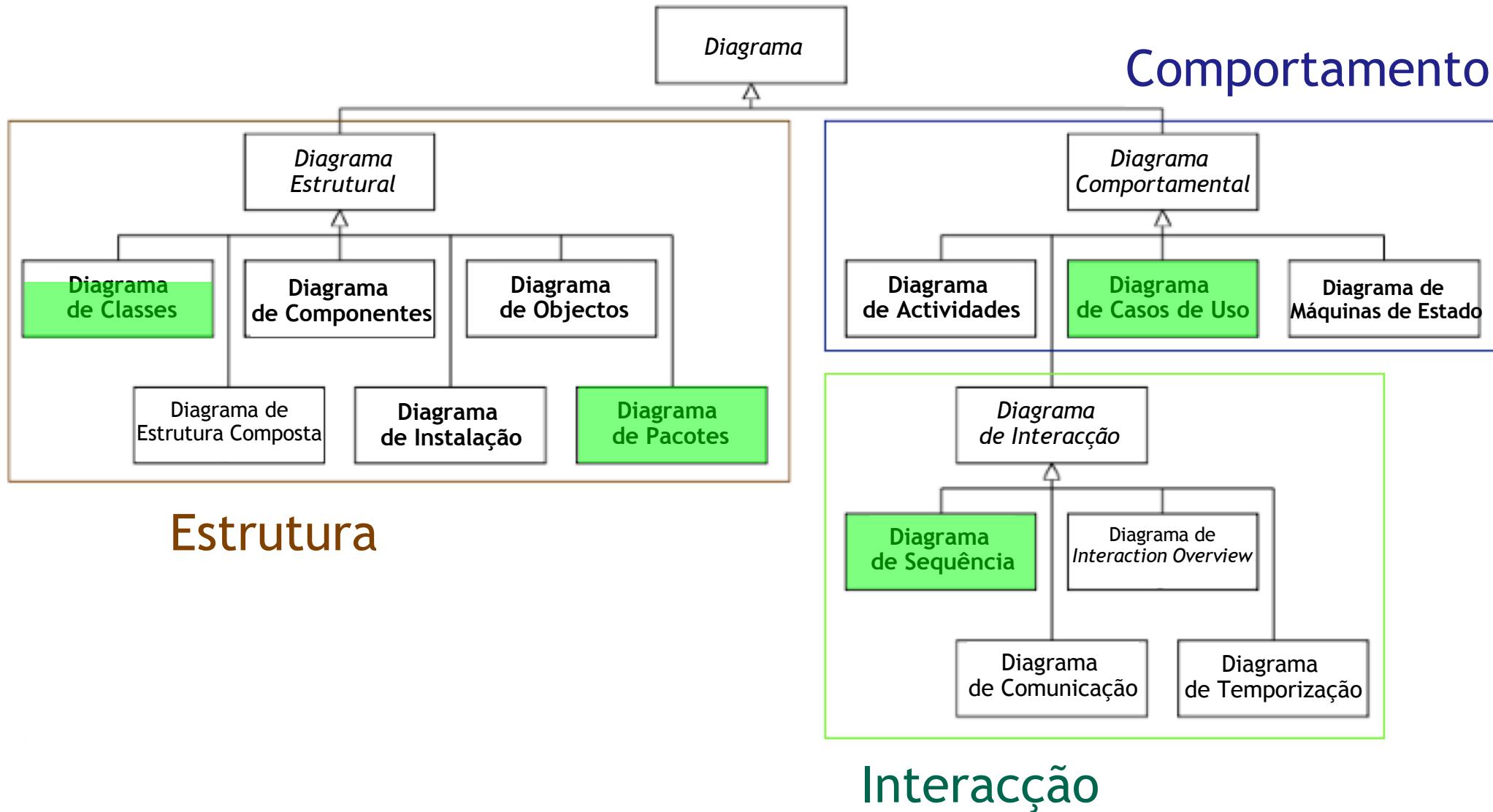


Desenvolvimento de Sistemas Software

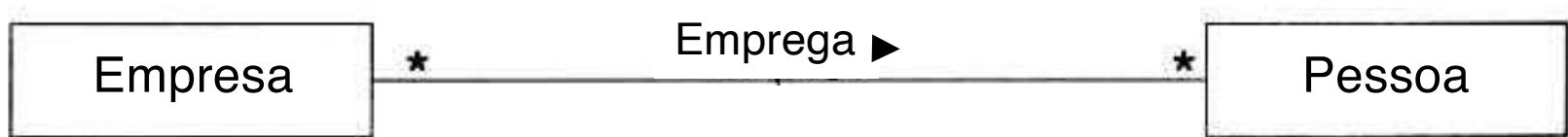
Aula Teórica 15

Modelação Estrutural / Diagramas de Classe III

Diagramas da UML 2.x



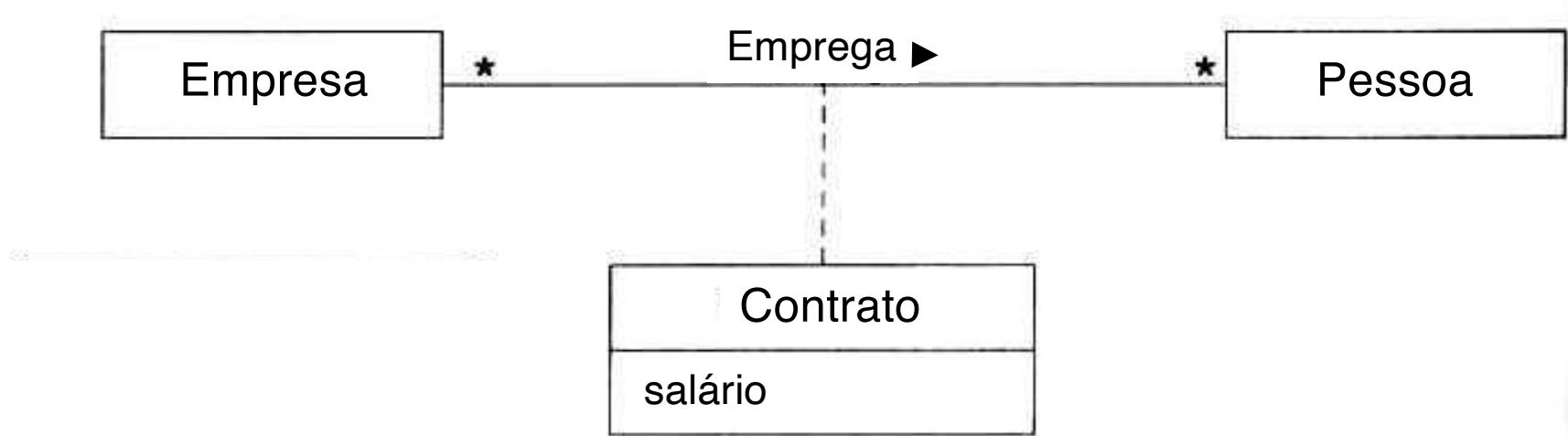
Classes de Associação



Contrato?

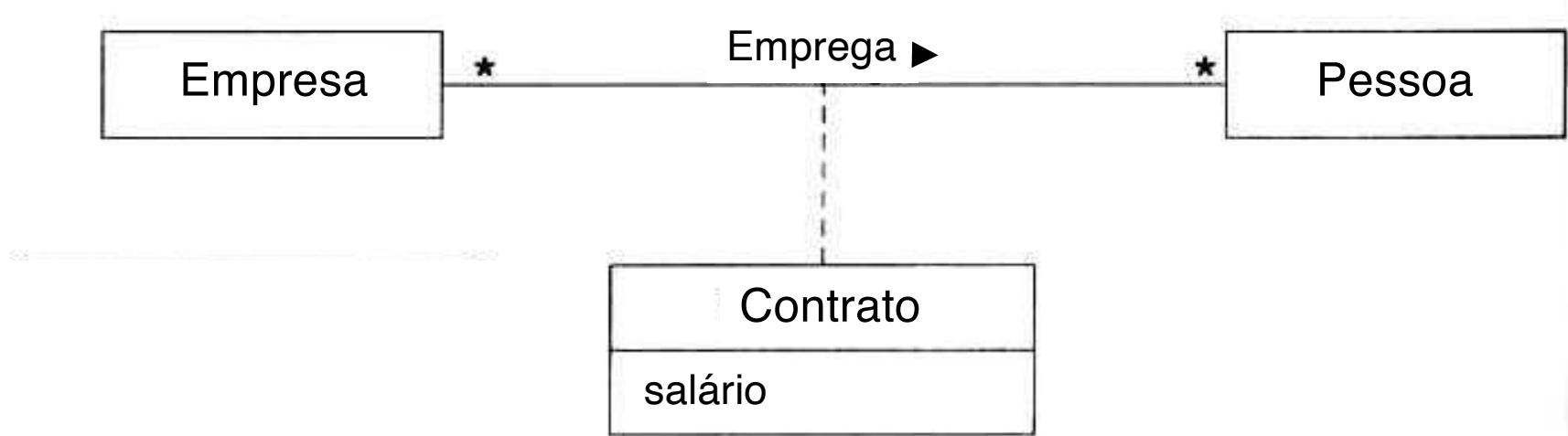
- A relação entre cada Empresa e cada um dos seus funcionários é caracterizada por um contrato.
- Cada Pessoa, pode ter estado contratada por várias Empresas e para cada uma há um contrato diferente.
- O Contrato não é característica da Empresa, nem da Pessoa, mas da relação entre ambas.

Classes de Associação



- A relação entre cada Empresa e cada um dos seus funcionários é caracterizada por um contrato.
- 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.

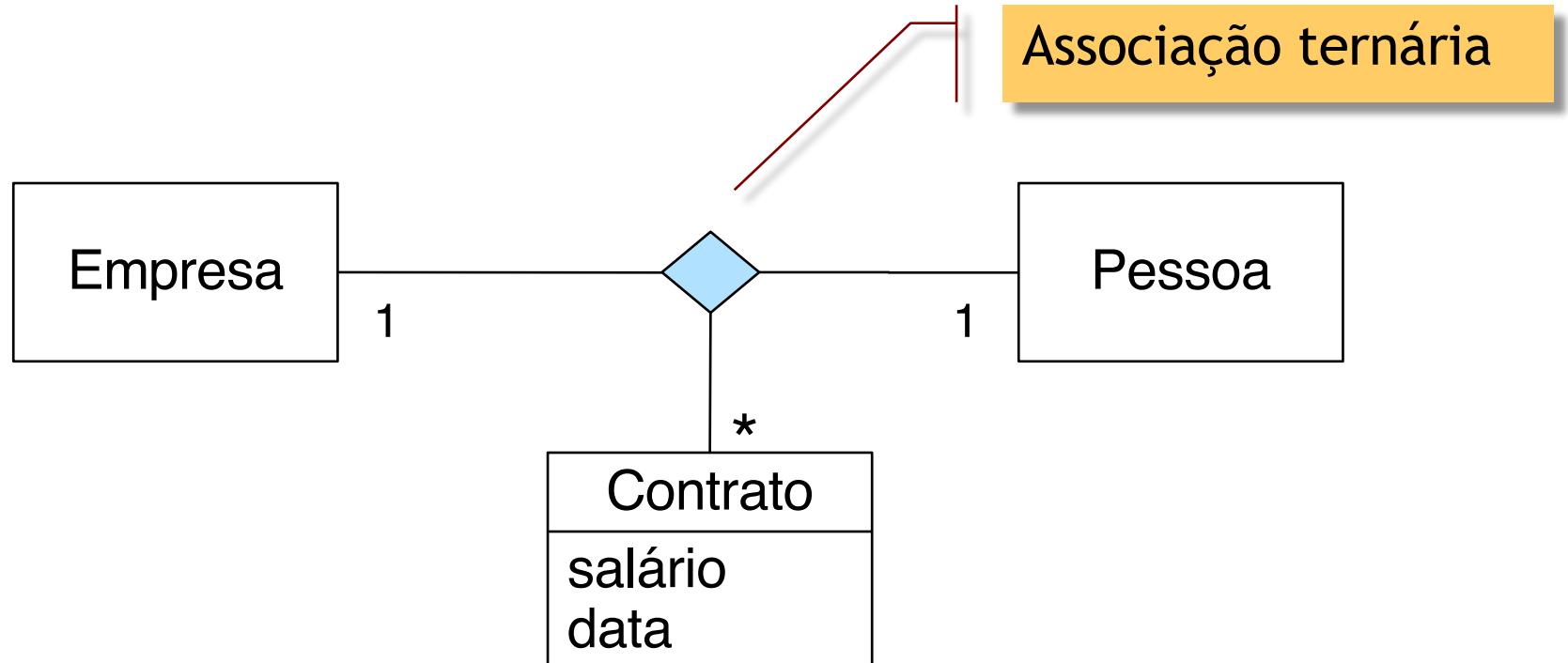
Classes de Associação



- A relação entre cada Empresa e cada um dos seus funcionários é caracterizada por um contrato.
- 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.
- mas...
Dois contratos diferentes com a mesma empresa?!

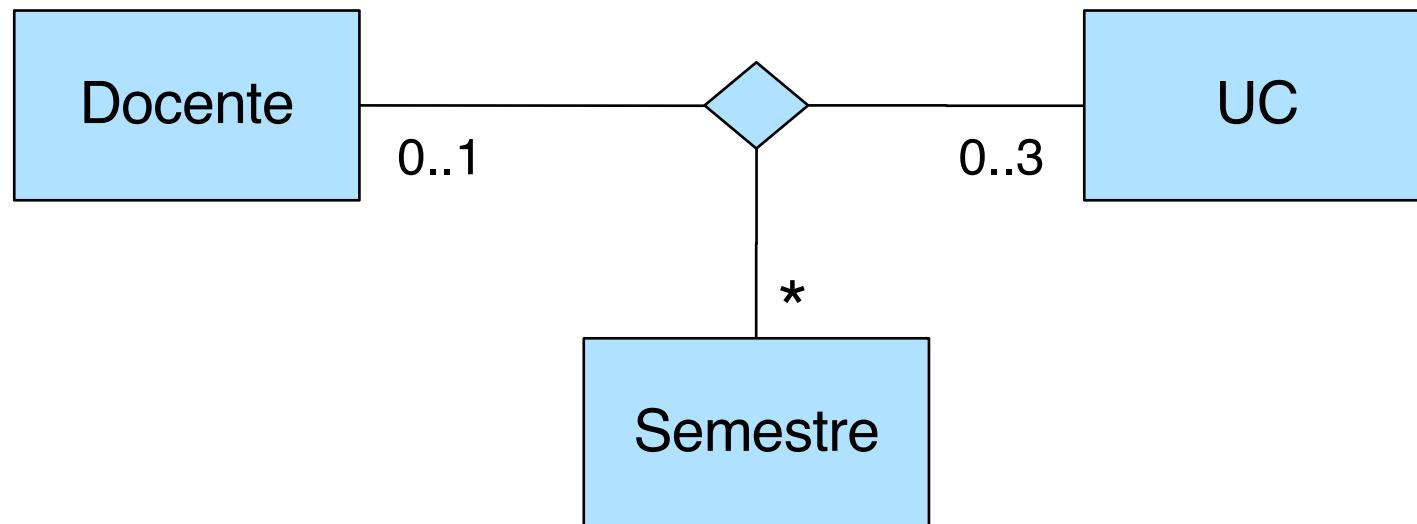
Associações n-árias

- 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.

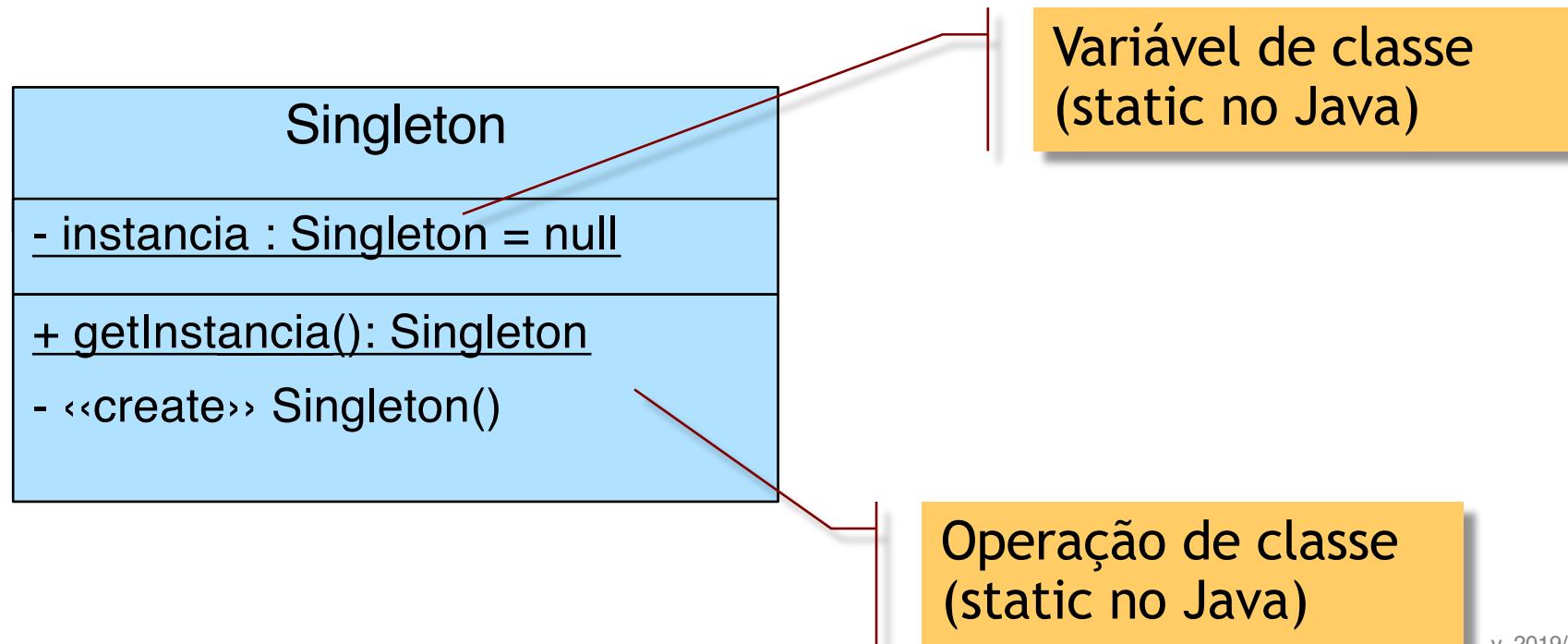
Associações n-árias - outro exemplo



- 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

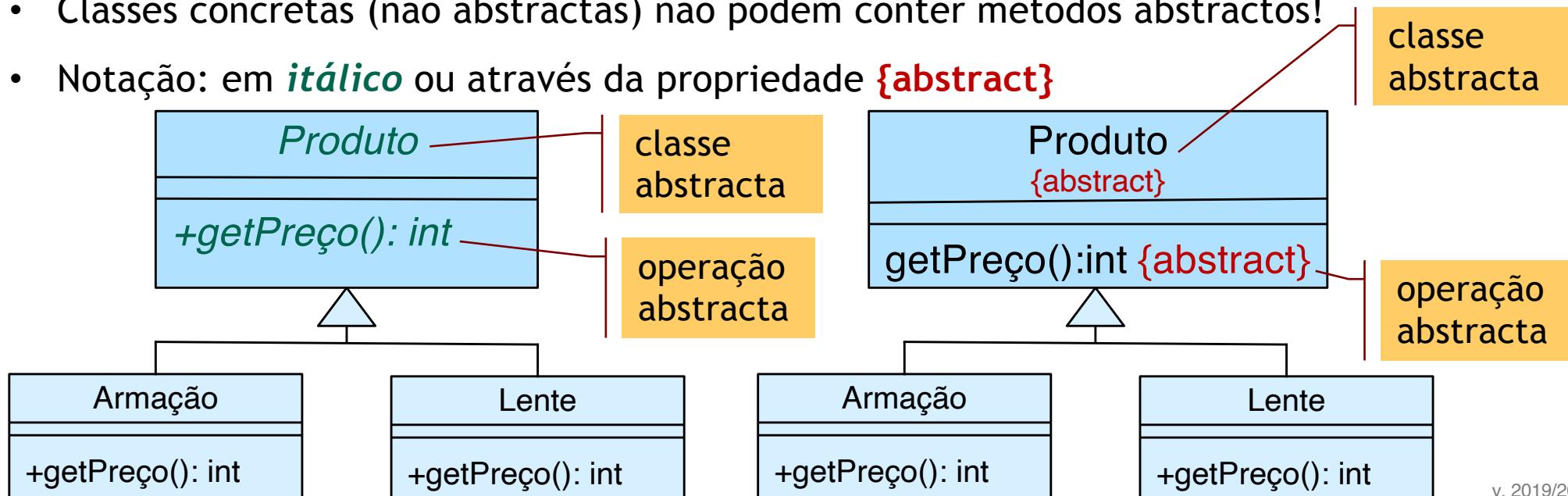
Operações e variáveis de classe

- Variáveis de classe são globais a todas as instâncias de uma classe.
- Métodos de classe são executados directamente pela classe e não pelas instâncias (logo, não tem acesso directo a variáveis/métodos de instância).
- São representados tal como variáveis/métodos de instância, mas sublinhados.
- Deve evitarse abusar de operações e variáveis e classe.



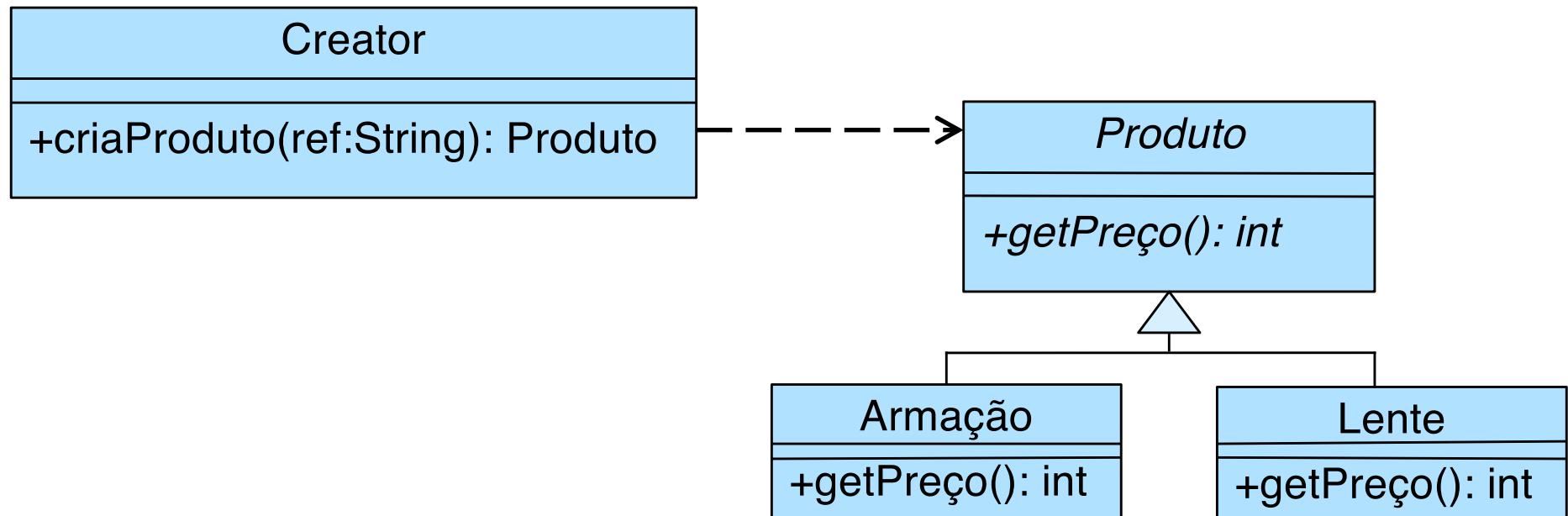
Classes abstractas

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

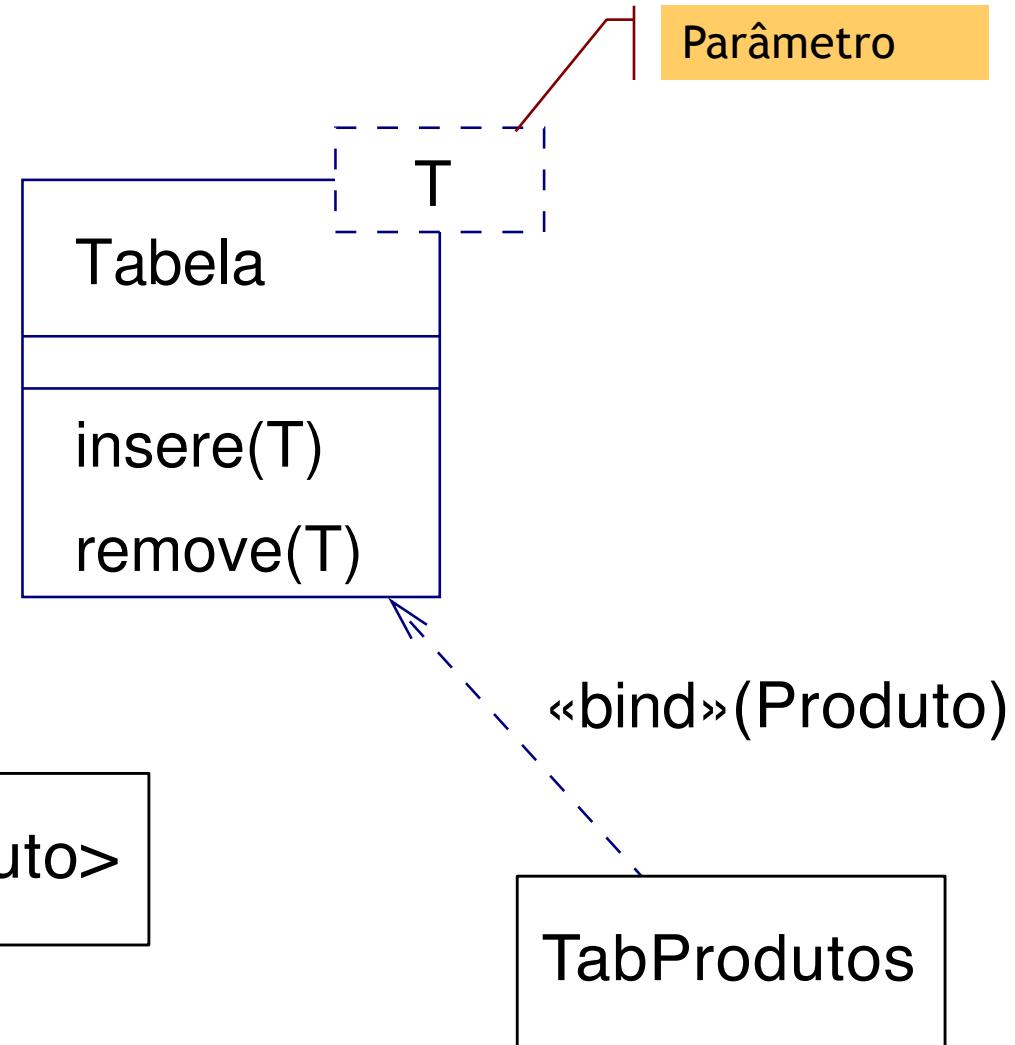


Exemplo - Factory method pattern

- O padrão arquitectural *Factory Method* utiliza-se quando temos uma API para criar objectos, mas apenas em tempo de execução vamos saber em concreto qual a classe que vamos querer instanciar.



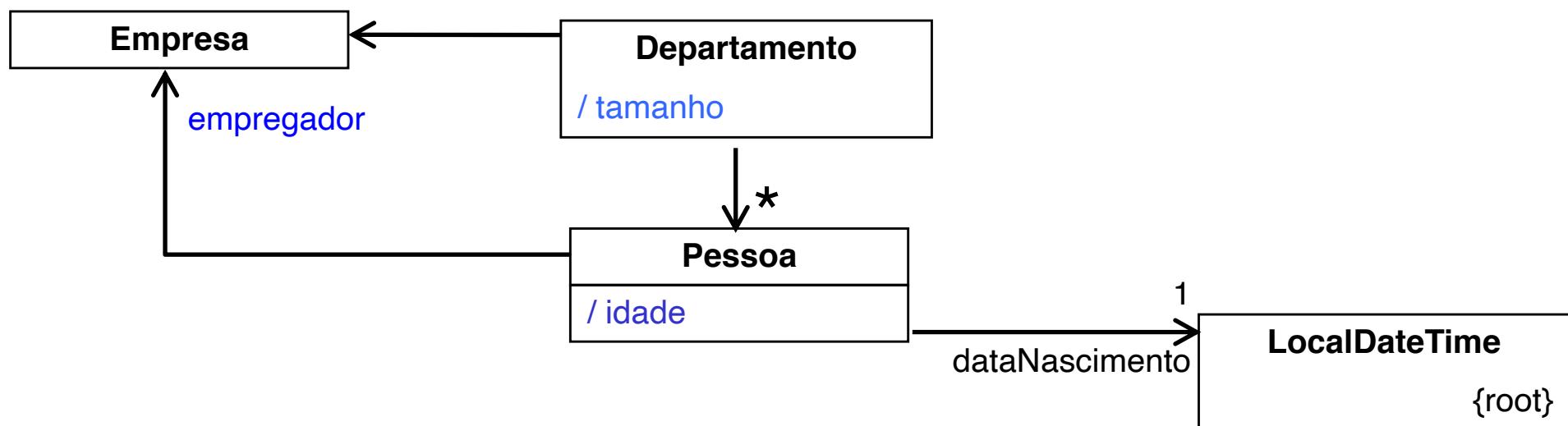
Classes parametrizadas (*Template classes*)



Em Java: Generics!

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 active

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

WorkerClass
{active}

- *Notação gráfica*



WorkerClass



WorkerClass

Classes leaf

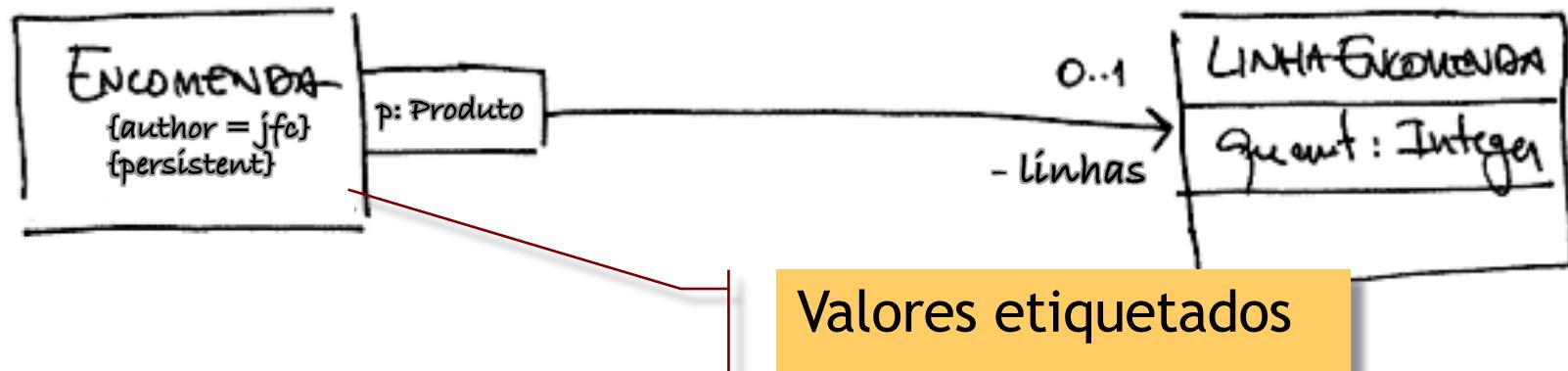
- Classes etiquetadas com a propriedade {leaf} não podem ser especializadas (classes final no Java).
 - Por exemplo, se o sistema contém uma classe que fornece serviços de encriptação, por motivos de segurança não é desejável que os métodos associados às operações dessa classe possam ser redefinidos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

CifraWorker

{leaf}

Mecanismos de extensibilidade

- “Tagged Values” (valores etiquetados)
- Estereótipos
- Restrições (“constraints”)
- Valores Etiquetados
 - Definem novas propriedades das “coisas”
 - Trabalham ao nível dos meta-dados

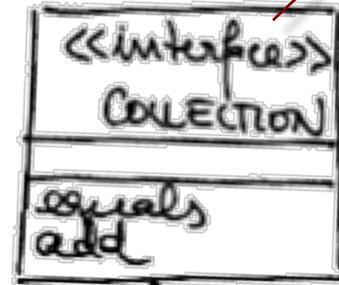


Mecanismos de extensibilidade

- **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

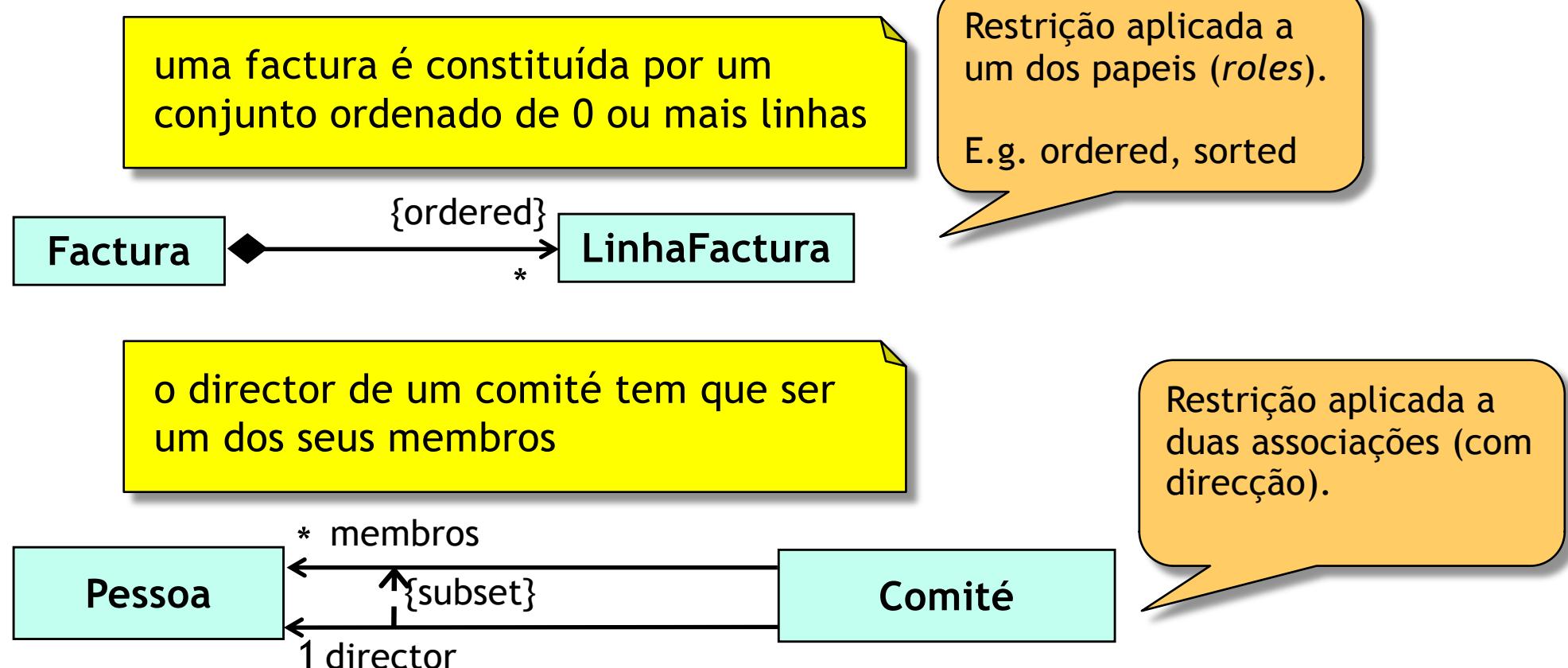
Estereótipo



Mecanismos de extensibilidade

- Restrições

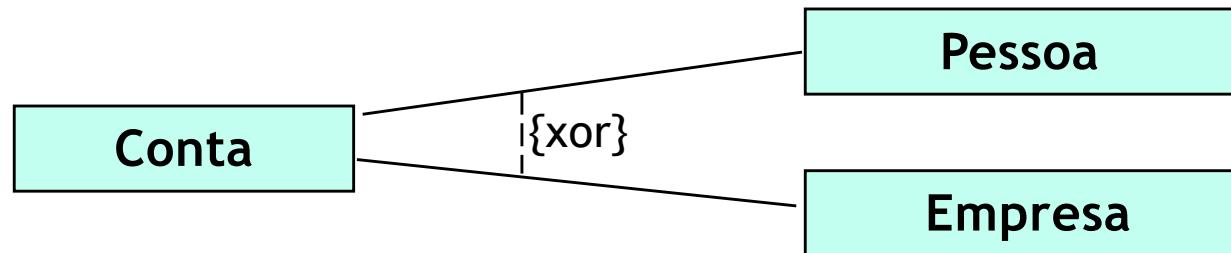
- Utiliza-se quando a semântica das construções diagramáticas do UML não é suficiente



Restrições às associações

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

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



- Vermos mais sobre restrições quando falarmos de OCL



Interfaces

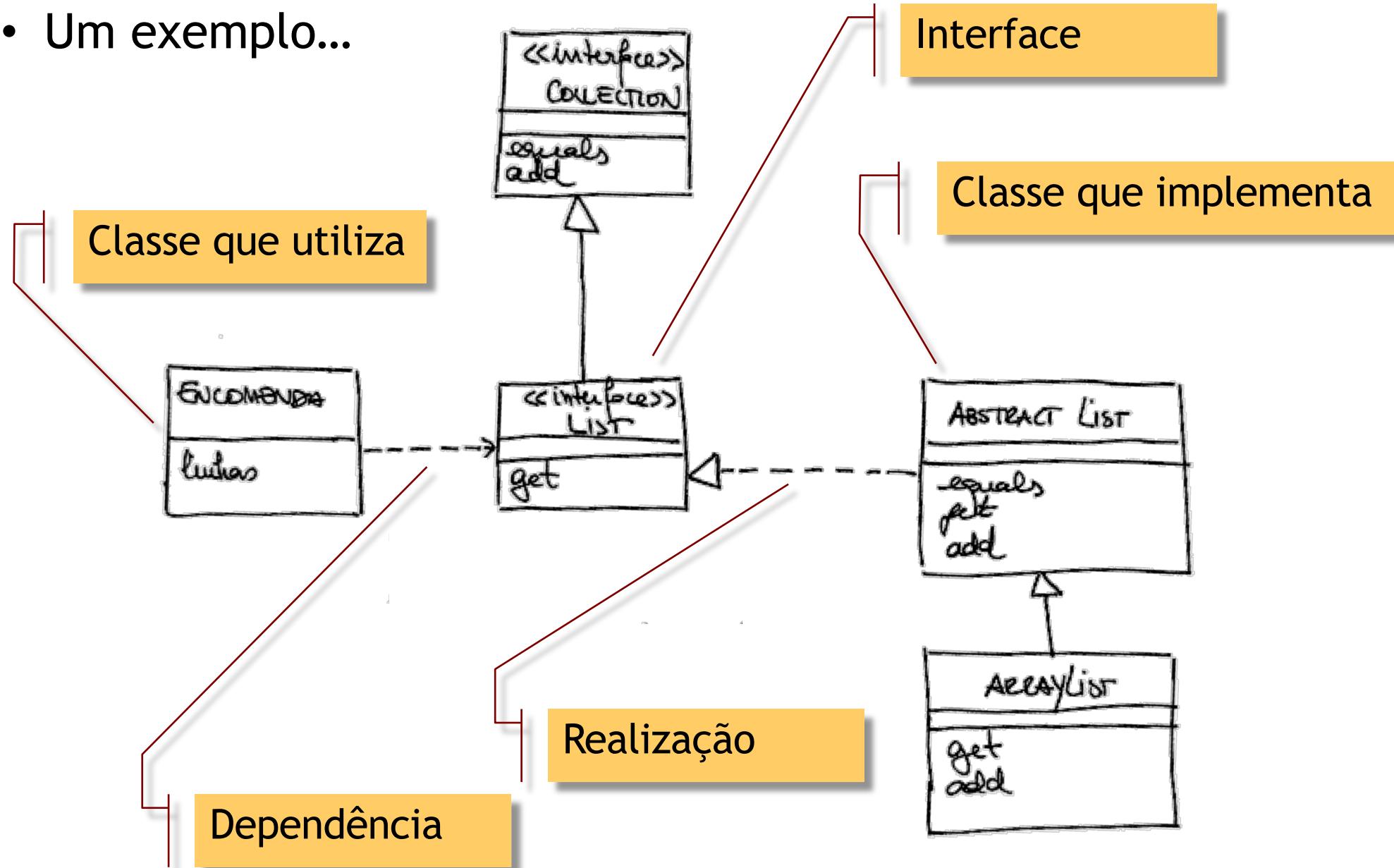
- Uma interface especifica um tipo abstracto - um conjunto de operações externamente visíveis que uma classe (ou componente, subsistema, etc.) deve implementar
- semelhante a classe abstracta só com operações abstractas e sem atributos nem associações
- separação mais explícita entre interface e (classes 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).

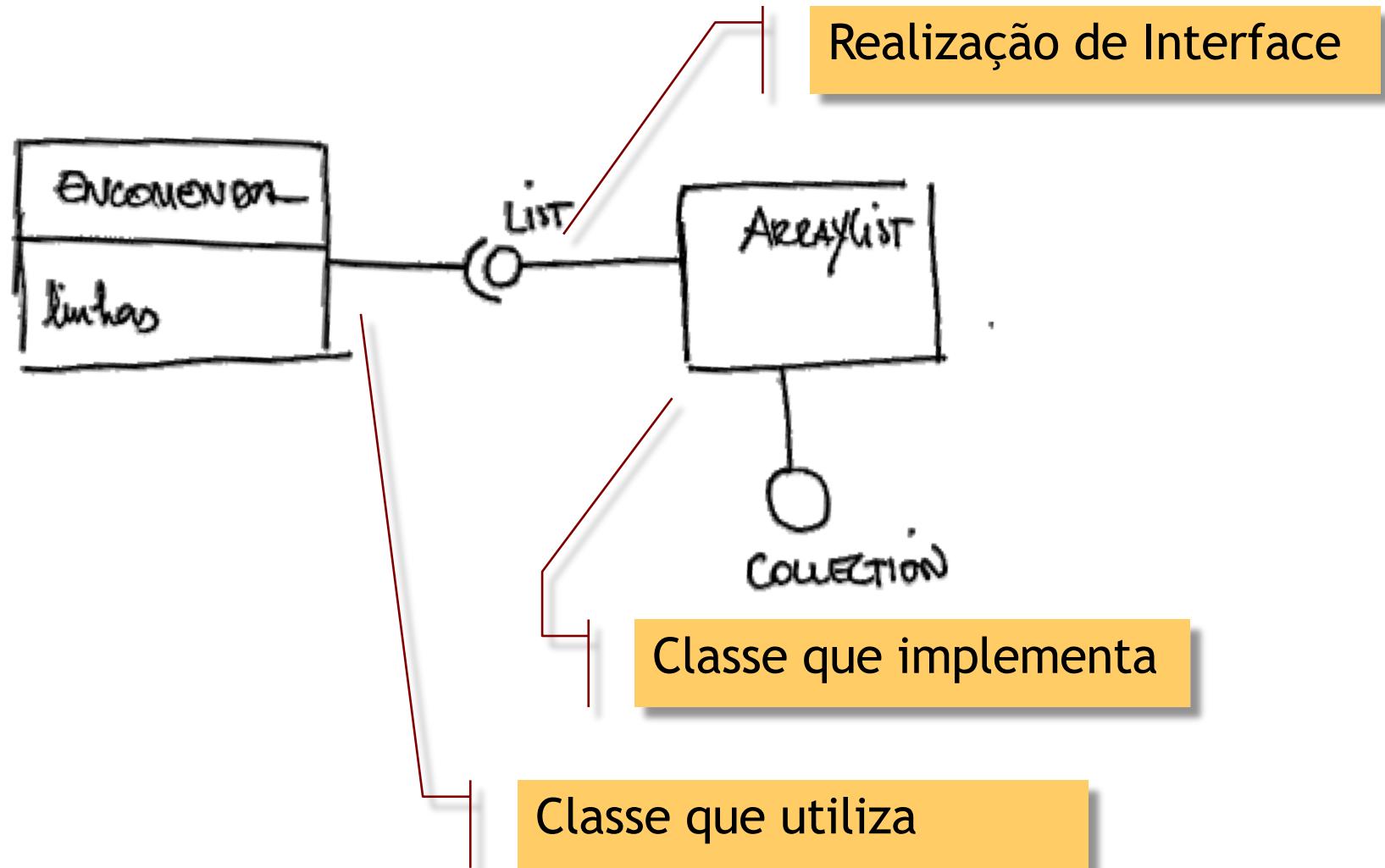
Interfaces

- Um exemplo...

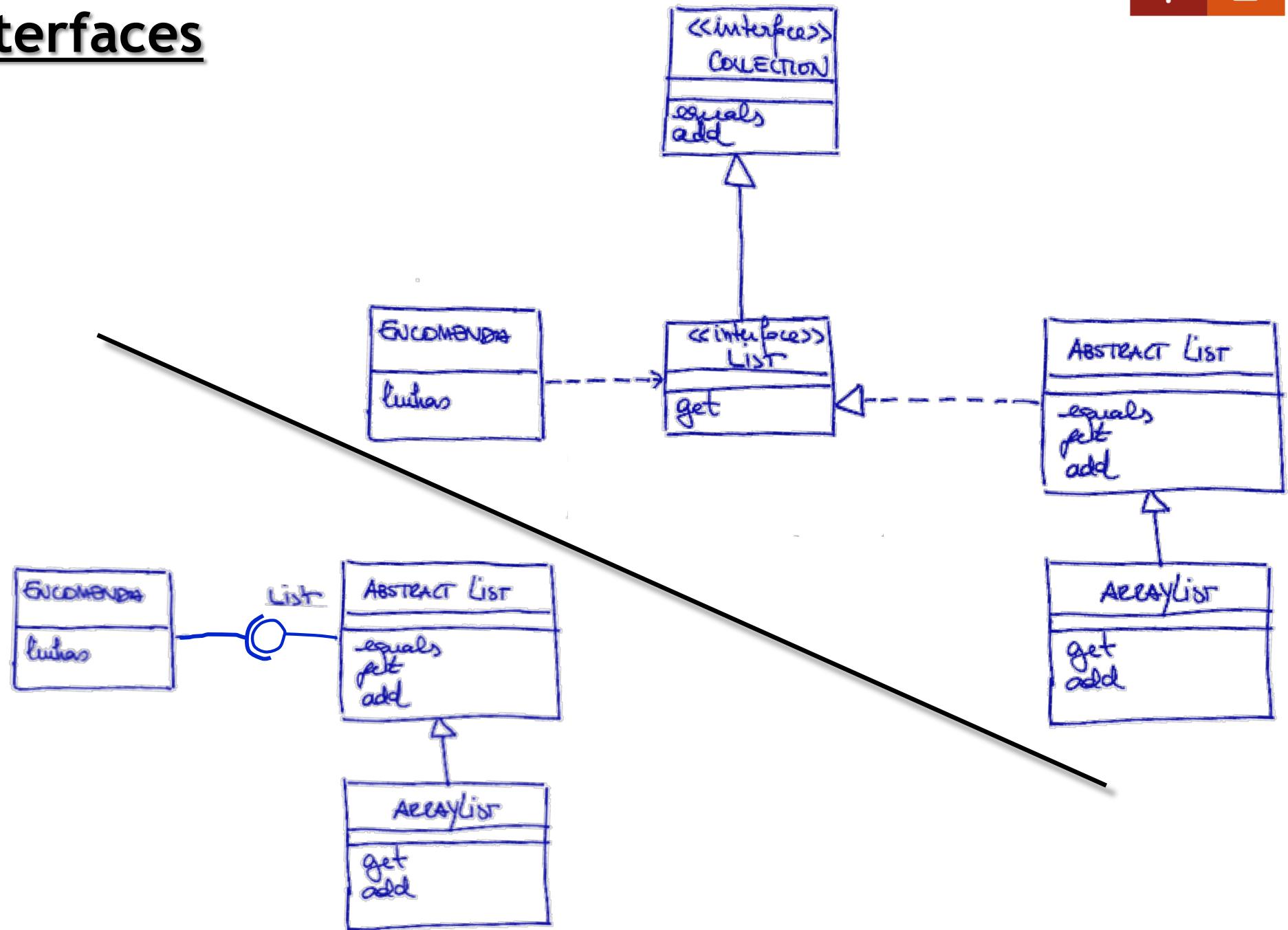


Interfaces

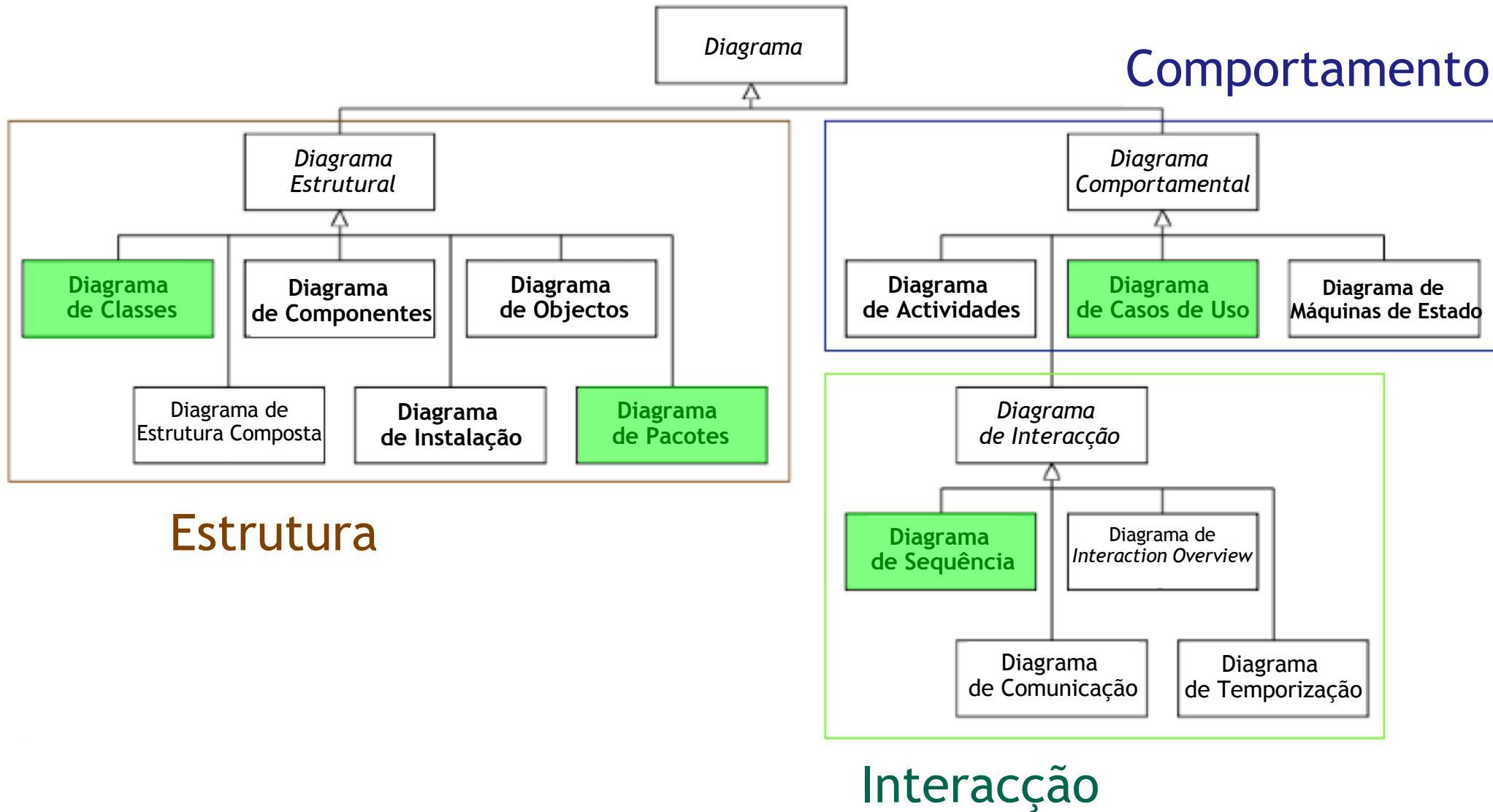
- Notação “lollipop” do UML 2.x



Interfaces



Diagramas da UML 2.x





Modelação Estrutural

Sumário

- Diagramas de Classe III
 - Classes de associação
 - 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