

Desenvolvimento de Sistemas Software



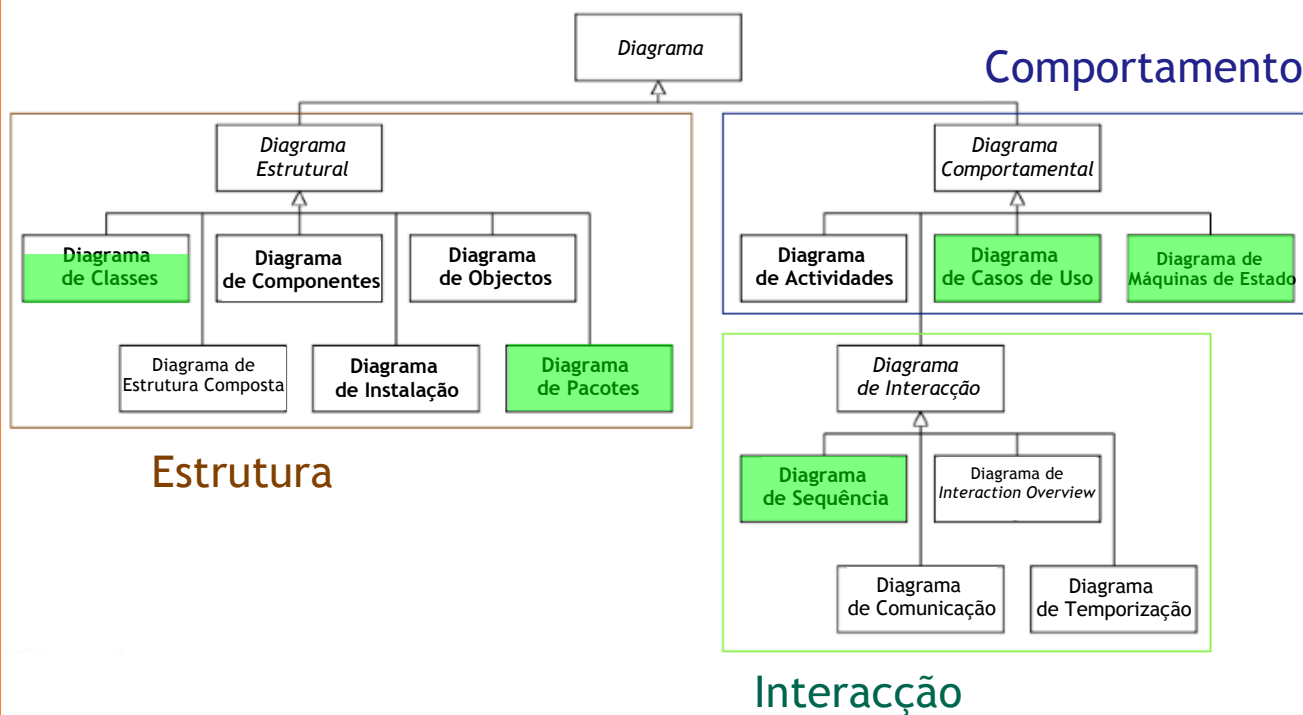
Aula Teórica 17

Modelação Estrutural / Diagramas de Classe III

v. 2017/18

342

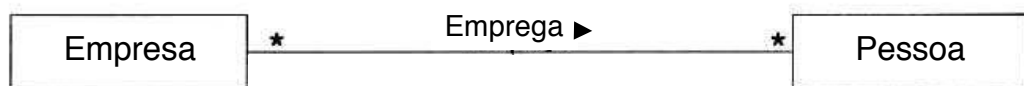
Diagramas da UML 2.x



v. 2017/18



Classes de Associação



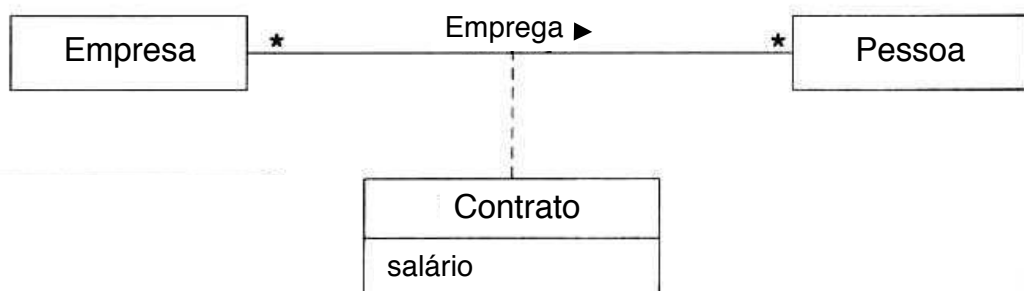
Contrato?

- 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.

v. 2017/18



Classes de Associação

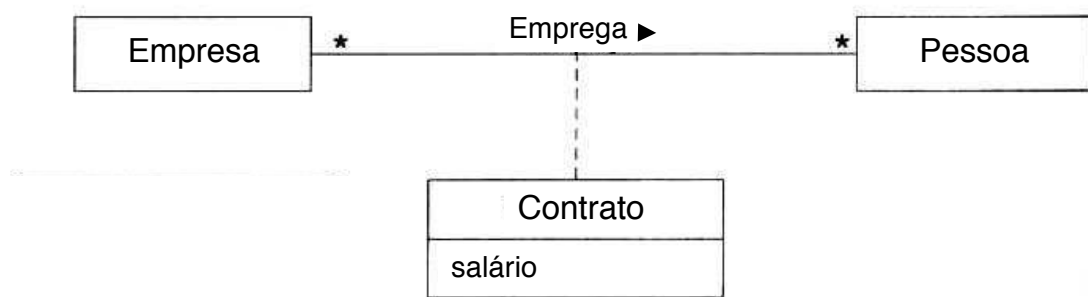


- A relação entre cada Empresa e cada um dos seus funcionários é caracterizada por um contrato.
- Cada Pessoa, pode ter estado contratada por várias Empresas e para cada uma há um contrato diferente.
- O Contrato não é característica da Empresa, nem da Pessoa, mas da relação entre ambas.

v. 2017/18



Classes de Associação



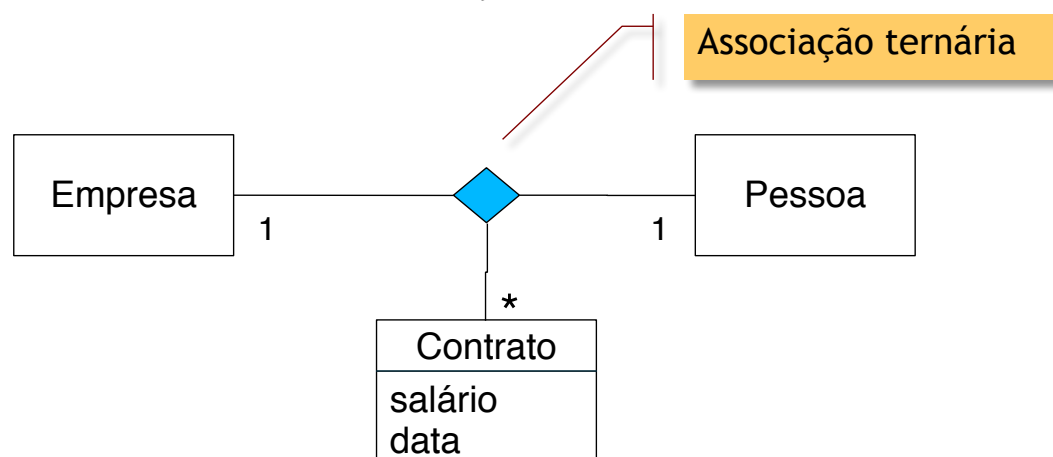
- A relação entre cada Empresa e cada um dos seus funcionários é caracterizada por um contrato.
- Cada Pessoa, pode ter estado contratada por várias Empresas e para cada uma há um contrato diferente.
- O Contrato não é característica da Empresa, nem da Pessoa, mas da relação entre ambas.
- mas...
Dois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

v. 2017/18



Associações n-árias

- A UML não se restringe a associações binárias:

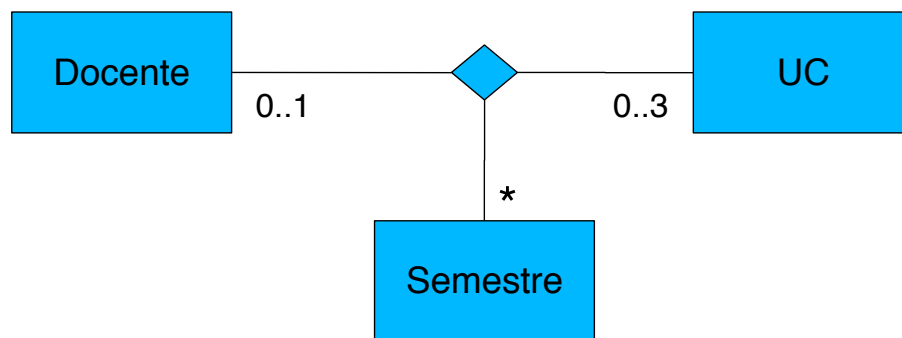


- Multiplicidades indicam quantos objectos existem para uma dada combinação de objectos das outras classes.
- Navegabilidade, agregação e qualificação **não são permitidos**.

v. 2017/18



Associações n-árias - outro exemplo



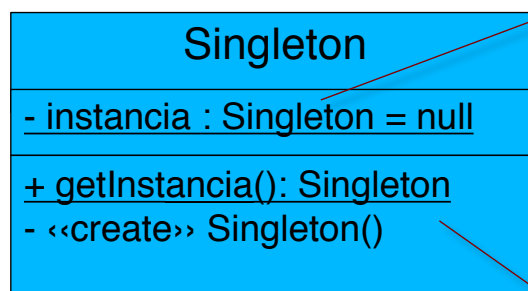
- Cada docente pode leccionar num semestre, no máximo, três UCs.
- Uma multiplicidade de zero invalida a combinação de objectos(!)
 - Não é possível ter uma associação entre uma UC e um Semestre sem indicar o Docente
 - Não é possível dizer que um Docente dá aulas num Semestre sem indicar, pelo menos,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

v. 2017/18



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 evitar-se abusar de operações e variáveis de classe.



Variável de classe
(static no Java)

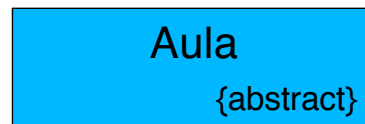
Operação de classe
(static no Java)

v. 2017/18



Classes abstractas

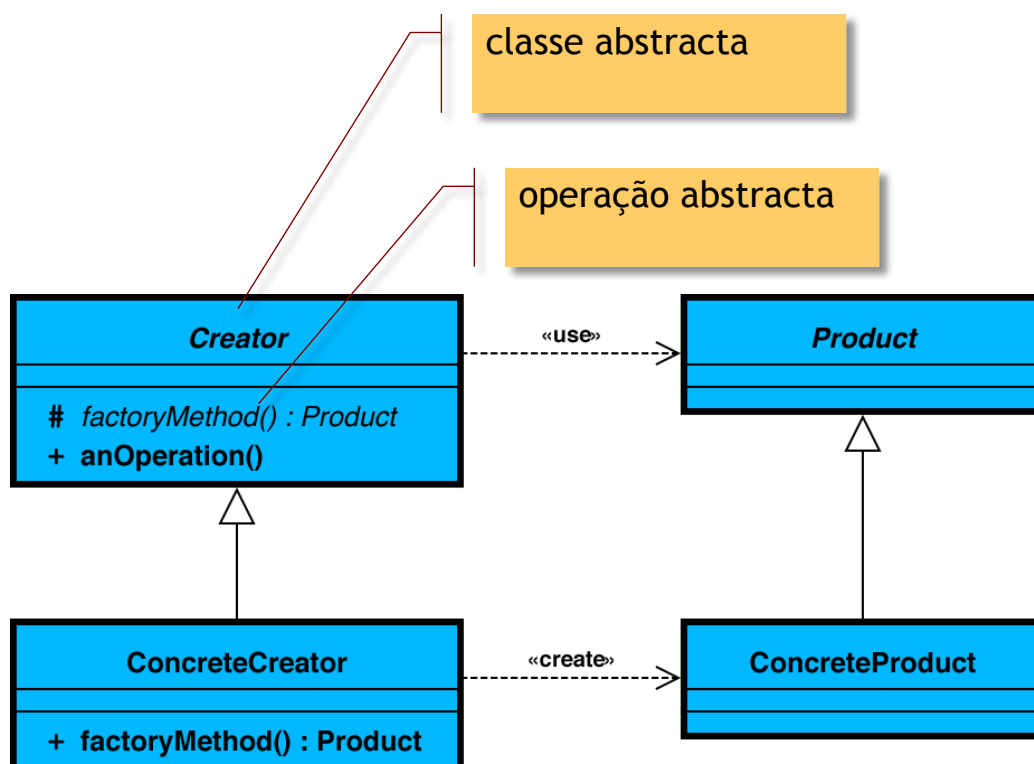
- Nem sempre ao nível da super-classe é possível saber qual deverá ser o método associado a uma operação.
- Quando se está a utilizar uma hierarquia de classes para representar sub-tipos, pode não fazer sentido permitir instâncias da super classe.
- Uma operação abstracta é uma operação que não tem método 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}.



v. 2017/18



Exemplo

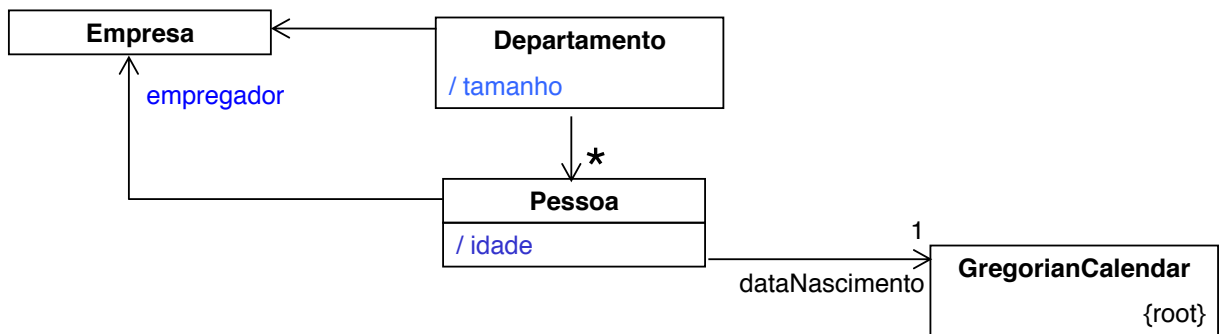


v. 2017/18



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.

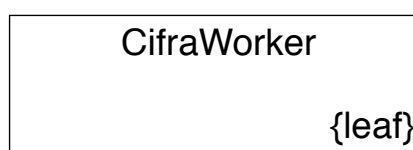


v. 2017/18



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).

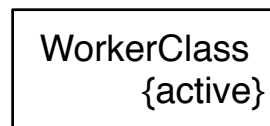


v. 2017/18



Classes active

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



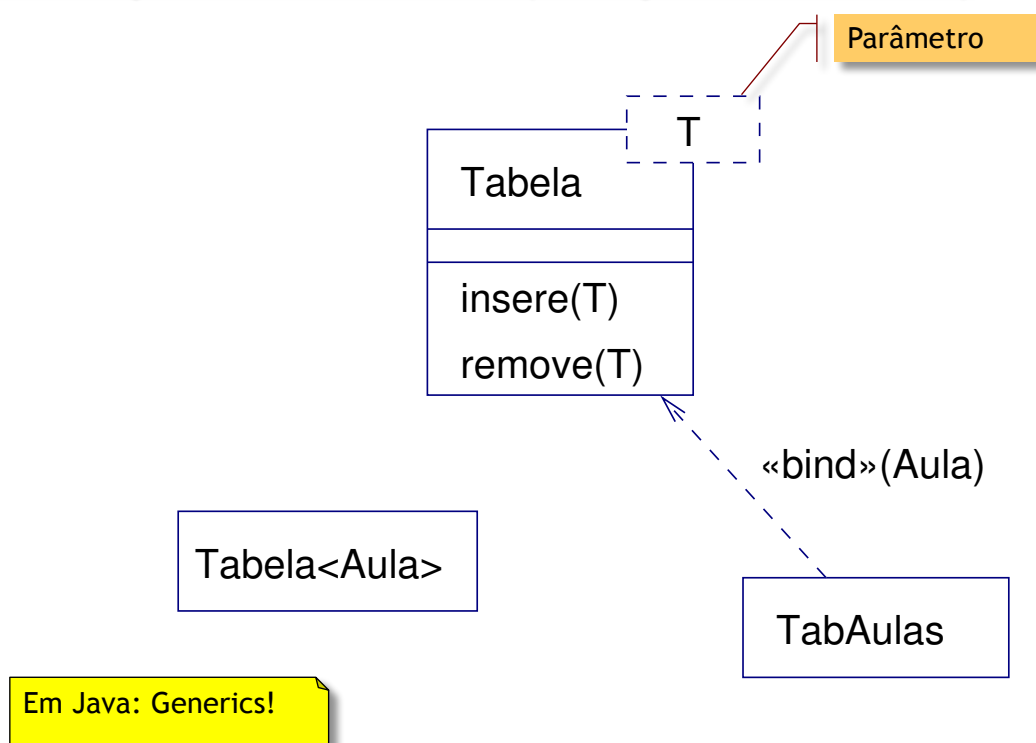
- Notação gráfica*



v. 2017/18



Classes parametrizadas (*Template classes*)

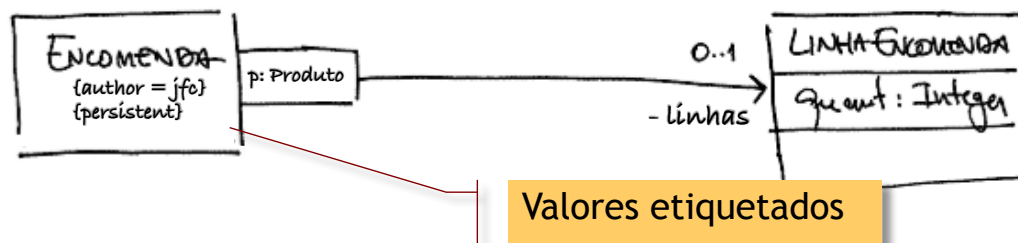


v. 2017/18



Mecanismos de extensibilidade

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

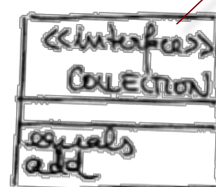


v. 2017/18



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

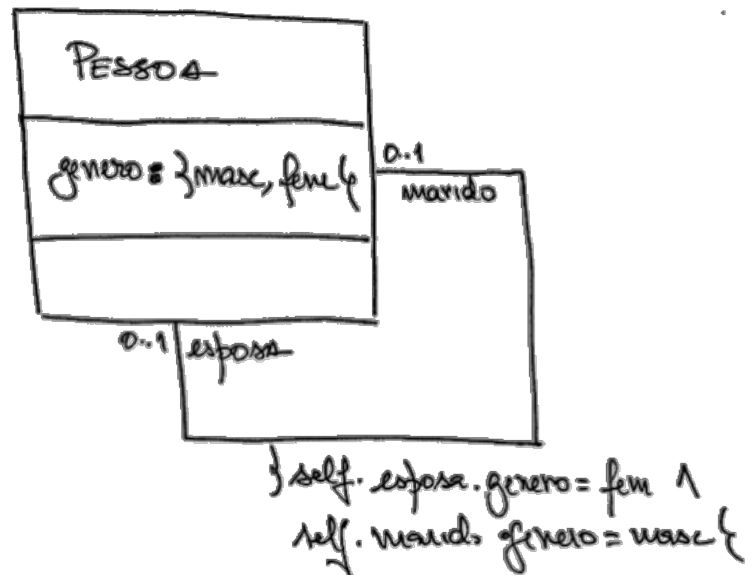
v. 2017/18



Mecanismos de extensibilidade

• Restrições

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

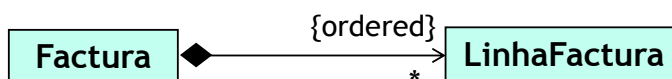


v. 2017/18



Restrições às associações

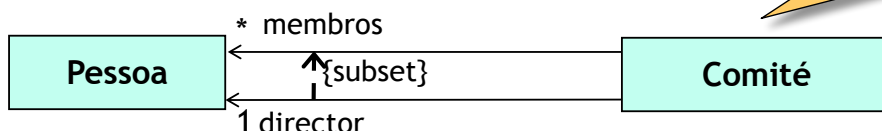
uma factura é constituída por um conjunto ordenado de 0 ou mais linhas



Restrição aplicada a um dos papeis (roles).

E.g. ordered, sorted

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



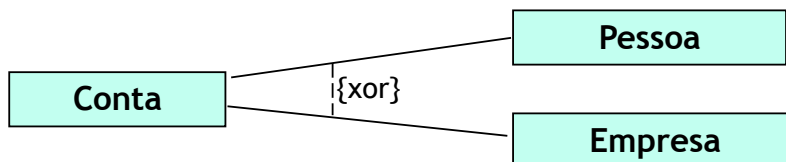
Restrição aplicada a duas associações (com direcção).

v. 2017/18



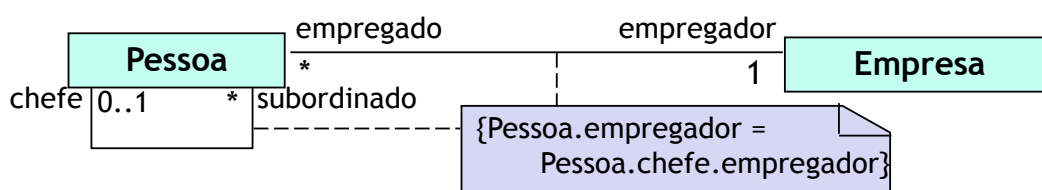
Restrições às associações

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.

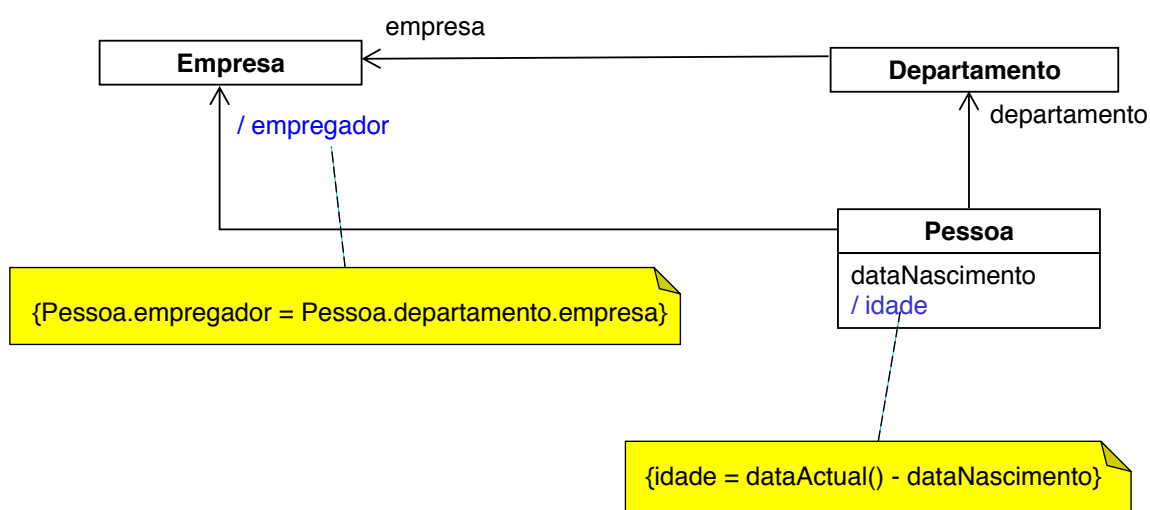
o empregador do chefe, é o empregador do subordinado



v. 2017/18



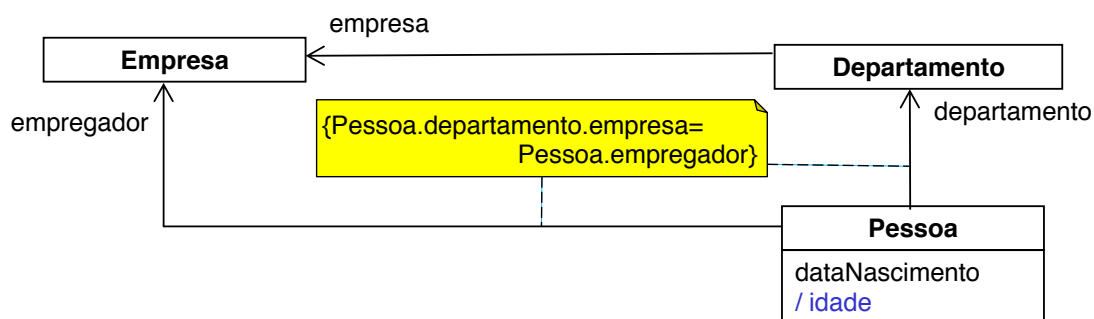
Exemplos de restrições



v. 2017/18



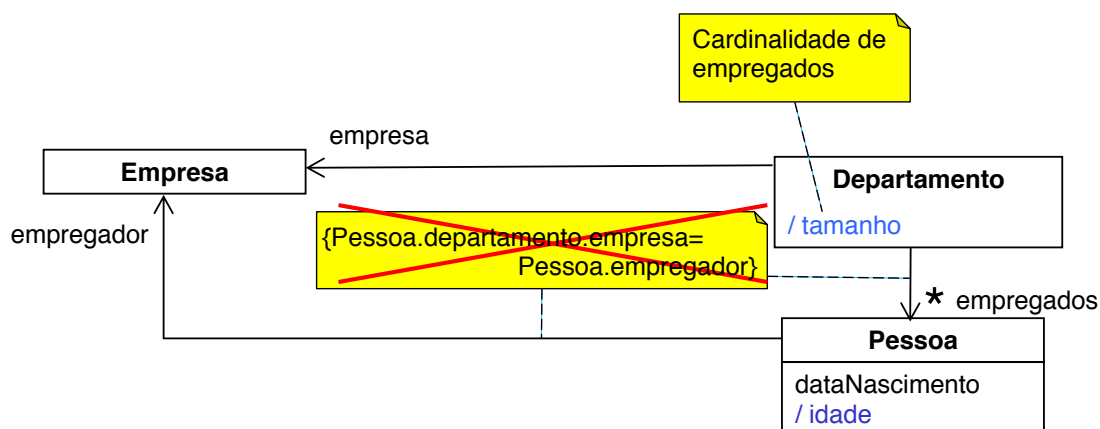
Exemplo restrições



v. 2017/18



Exemplos de atributos derivados



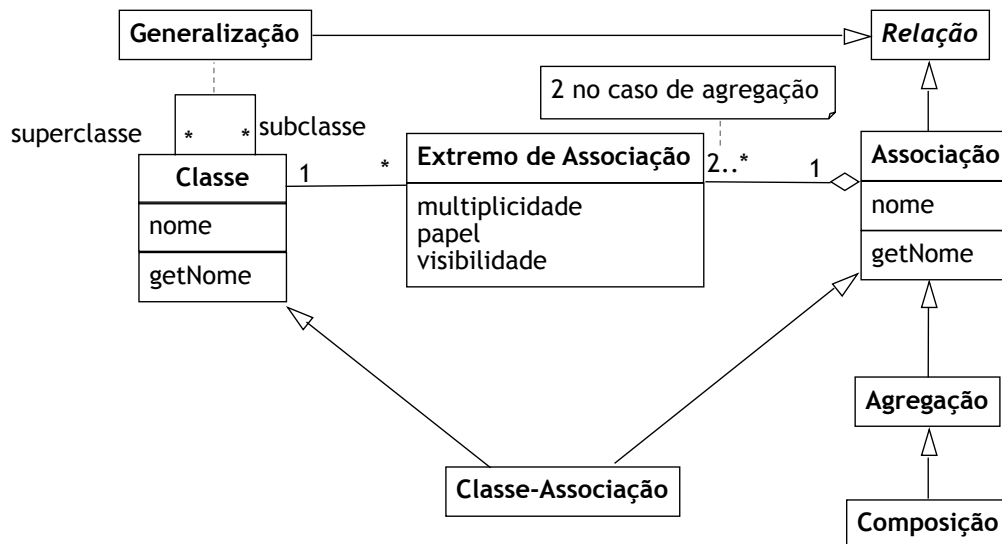
- Restrições tem que respeitar navegabilidade
- Alteração na navegabilidade impossibilita esta restrição
- Onde colocar restrição e qual?

v. 2017/18



Meta-modelo do UML (relações entre classes)

- É possível descrever em UML a semântica dos diagramas



v. 2017/18



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)

v. 2017/18



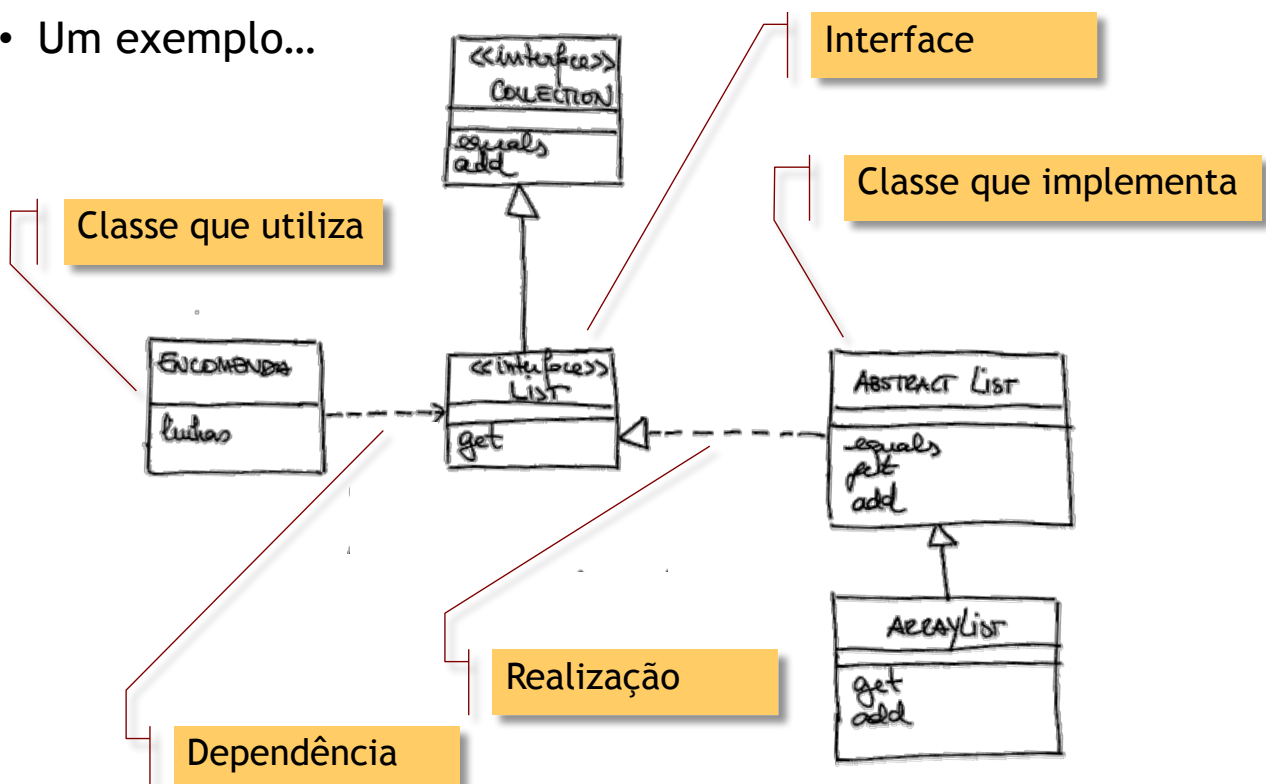
Interfaces

- 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

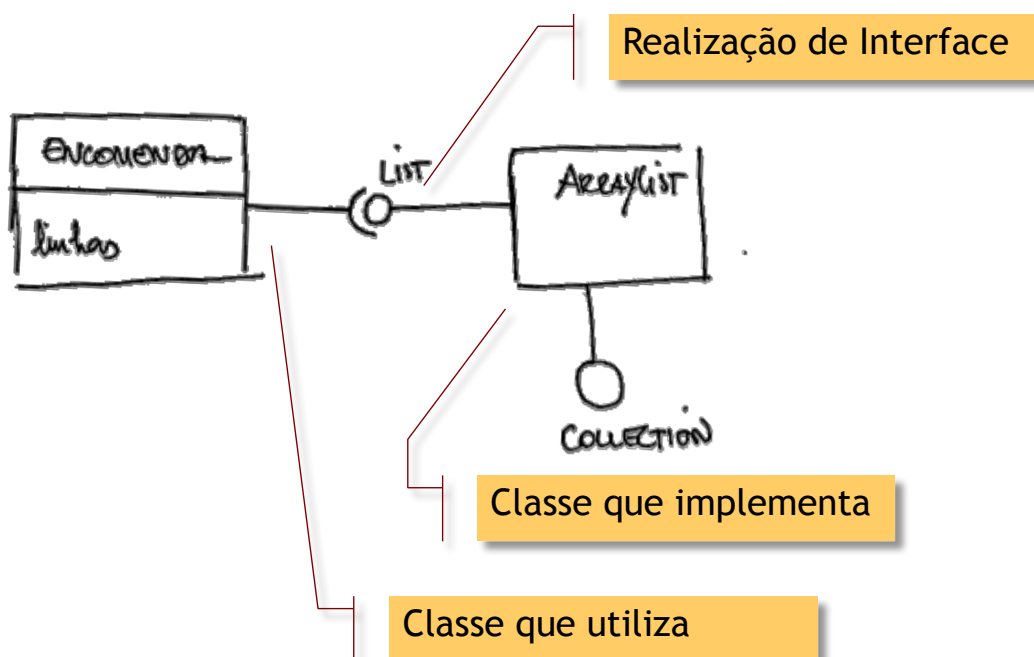
- Um exemplo...





Interfaces

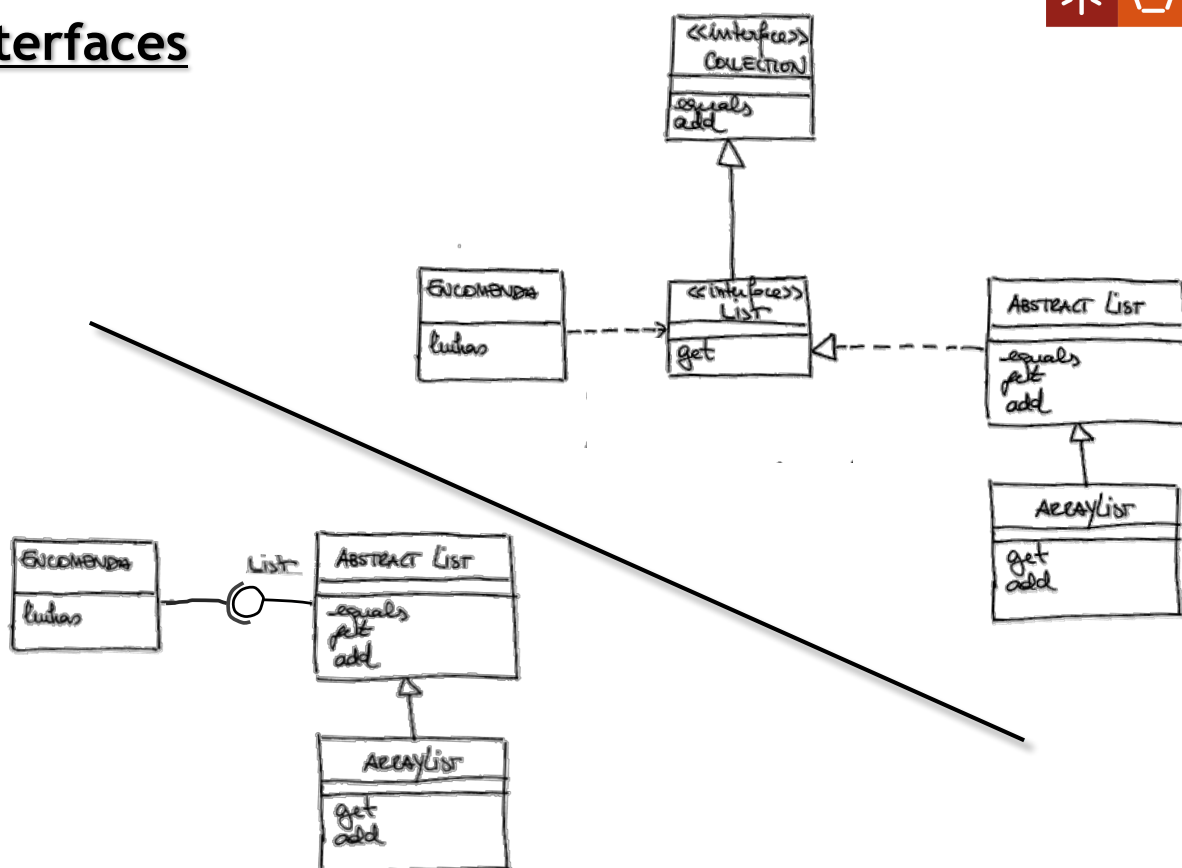
- Notação “lollipop” do UML 2.x



v. 2017/18



Interfaces



v. 2017/18



O diagrama ilustra as seguintes entidades e suas relações:

- Generalização**: Representa a relação entre uma **Classe** (superclasse) e uma **Classe-Associação** (subclasse).
- Classe**: Possui atributos **nome** e **getNome**.
- Extremo de Associação**: Possui atributos **multiplicidade**, **papel** e **visibilidade**.
- Associação**: Possui atributos **nome** e **getNome**. Está associada à **Classe** com multiplicidade **1** e à **Classe-Associação** com multiplicidade **2..***. O papel na associação com a classe é **- myassoc**.
- «interface» IAssociação**: Possui o método **getNomeAssoc**.
- Classe-Associação**: Implementa a interface **IAssociação** e possui os métodos **getNome** e **getNomeAssoc**. Está associada à **Associação** com multiplicidade **1** e à **Classe** com multiplicidade *****.
- Agregação**: Representa a relação entre a **Classe-Associação** e a **Associação**.
- Composição**: Representa a relação entre a **Classe-Associação** e a **Associação**.

Outros detalhes do diagrama:

- Um retângulo amarelo indica: **Redireccionado para myassoc**.
- Um retângulo amarelo indica: **Herdado**.
- Um retângulo amarelo indica: **2 no caso de agregação**.

v. 2017/18



O diagrama apresenta a hierarquia dos tipos de diagramas de UML, organizados em três categorias principais: Estrutura, Comportamento e Interação.

- Diagrama** (Raiz)
 - Diagrama Estrutural** (Categoria: Estrutura)
 - Diagrama de Classes
 - Diagrama de Componentes
 - Diagrama de Estrutura Composta
 - Diagrama de Objectos
 - Diagrama de Instalação
 - Diagrama de Pacotes
 - Diagrama Comportamental** (Categoria: Comportamento)
 - Diagrama de Actividades
 - Diagrama de Casos de Uso
 - Diagrama de Máquinas de Estado
 - Diagrama de Interação** (Categoria: Interação)
 - Diagrama de Sequência
 - Diagrama de Comunicação
 - Diagrama de Interaction Overview
 - Diagrama de Temporização

v. 2017/18



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