

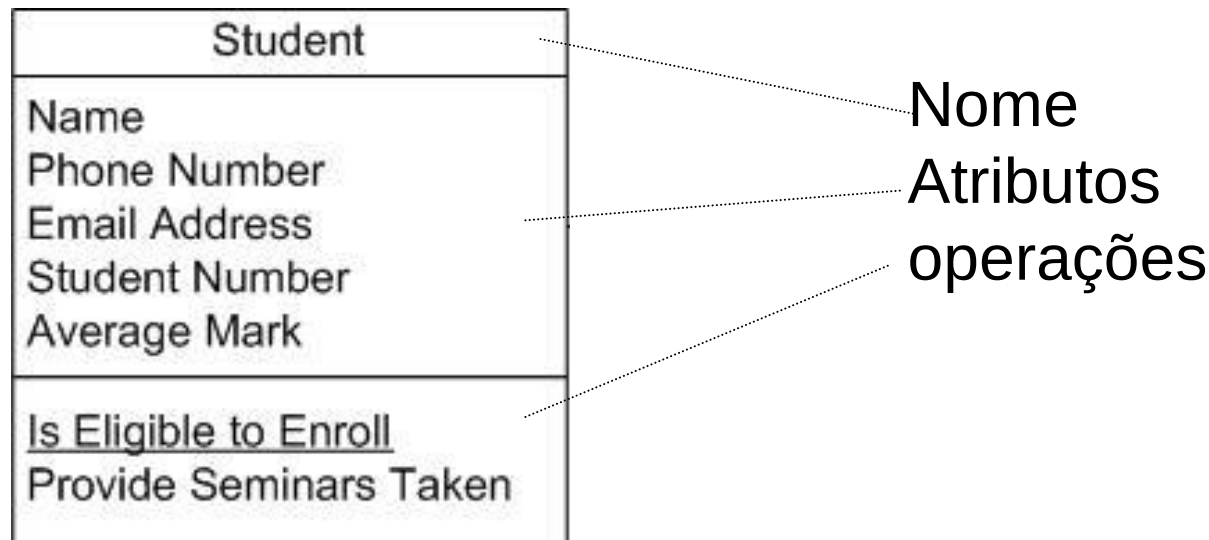
Métodos de Desenvolvimento de Software (MDS) 2011/2012

Miguel Goulão
mgoul@fct.unl.pt
<http://ctp.di.fct.unl.pt/~mgoul/>

Classes

2

- Uma classe descreve um conjunto de objectos que partilham os mesmos atributos, operações, associações e semântica



Nome da classe

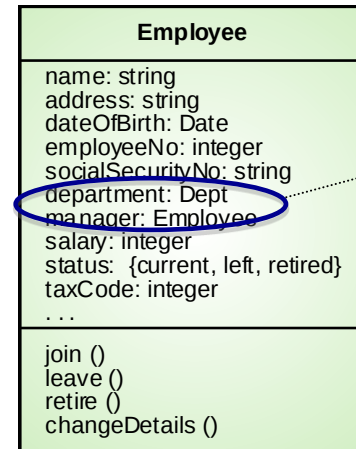
3

□ Identidade

- ▣ Substantivo ou termo substantivado curto do vocabulário do domínio

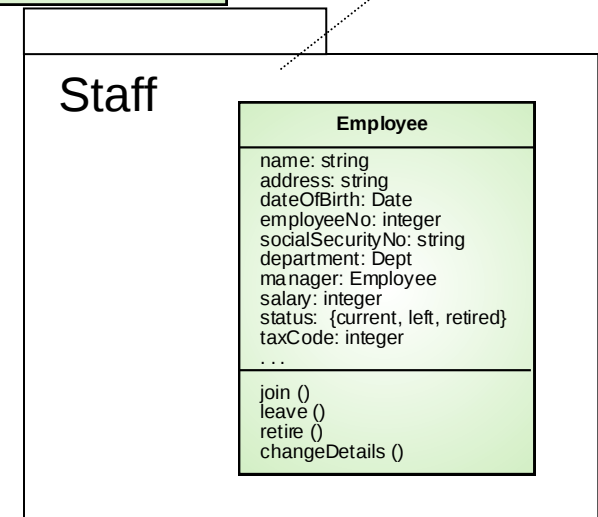
□ Nome simples

- ▣ Prefixado pelo nome do pacote, quando necessário, que define o caminho (path) onde a classe está definida (Staff::Employee);
- ▣ Classes diferentes com o mesmo nome são possíveis, desde que estejam em pacotes distintos



Que representam?
(classe de desenho!)

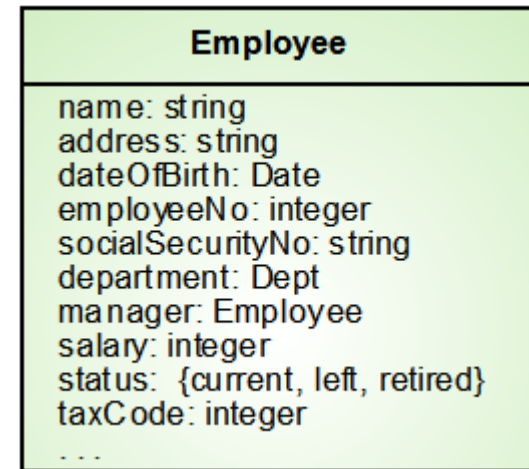
Pacote



Atributos

4

- Propriedade de uma classe
 - ▣ substantivo
- Descreve um conjunto de valores que uma instância da propriedade pode ter

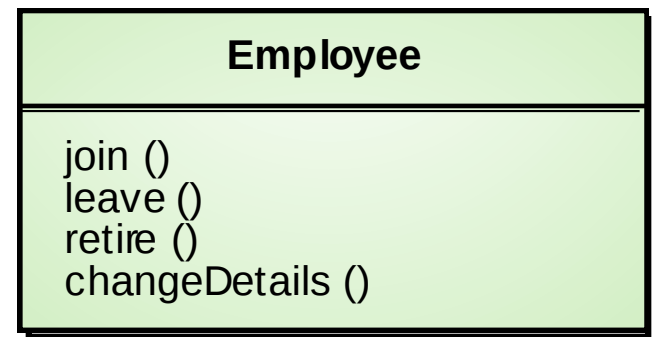


Operações escondidas

Operações

5

- Abstracção de algo que o objecto pode levar a cabo
 - ▣ Descreve-se com um verbo que representa comportamento da classe
- Partilhado por todos os objectos da classe



Atributos escondidos

Taxonomia das operações

6

□ Construtores

- Criam um novo objecto (atribuem espaço de memória a um objecto)

□ Selectores

- Retornam um valor (contido ou referido por um objecto)

□ Modificadores

- Provocam uma mudança de estado
- Mudanças de estado podem referir-se à mudança de valor de um ou mais atributos

□ Destrutores

- Destroem o objecto corrente (libertando espaço de memória)

Classes bem-formadas

7

- Representam uma abstracção de uma parte do problema
- Encapsulam um conjunto de responsabilidades bem definidas
- Oferecem uma separação clara entre a especificação e a implementação
 - E entre o que é externamente visível e o que deve ser oculto

Diagrama de classes do domínio

8

- Um Diagrama de Classes do Domínio contém o conjunto de classes base (classes do tipo entidade) do problema
- Este diagrama será mais tarde estendido para contemplar outros tipos de classes e dependências entre elas

Técnicas de identificação de classes

9

- Extracção de nomes (substantivos e nomes adjetivados)
 - ▣ Bom para identificar classes
 - ▣ Mau para identificar seus requisitos (características e relações)
- Identificação de responsabilidades
 - ▣ Exige bom conhecimento do domínio

Extracção de nomes

10

□ Exemplo:

- Os **clientes** do **banco** podem debitar e creditar **montantes** nas suas **contas**, ou pedir o **saldo** corrente. Estas operações são completadas usando as **caixas automáticas** (Multibanco) ou ao **balcão**. As transacções numa conta fazem-se por **cheque**, **ordens de pagamento**, ou então usando as **caixas automáticas** com um **cartão**. Há dois tipos de **contas**: à **ordem** e a **prazo**. Uma **conta a prazo** dá **juros** e não pode ser acedida através das **caixas automáticas**.
- Candidatos a classes: **cliente**, **banco**, **montante**, **conta**, **saldo**, **caixa automática**, **balcão**, **cheque**, **ordem de pagamento**, **cartão**, **conta à ordem**, **conta a prazo**, **juro**
 - Substantivos... quase todos, mas mesmo assim...
- Qual a lista final?

Identificação de responsabilidades

11

- Uma responsabilidade é uma obrigação (contrato) de uma classe
- Passos
 - ▣ Identificar conjuntos de classes que cooperam para atingir um comportamento específico
 - ▣ Identificar responsabilidades e, conseqüentemente as operações e atributos que cada classe deve conter para levar a cabo as suas responsabilidades
 - ▣ Dividir classes com demasiadas responsabilidades ou juntar classes demasiado simples
- Fontes
 - ▣ Enunciado do problema, Diagramas de Use Cases com os seus cenários, Diagramas de Actividades

Identificação de responsabilidades (cont.)

12

- Responsabilidade: é um contrato ou uma obrigação de uma classe.
 - ▣ Ex.: Uma classe TemperatureSensor é responsável por medir a temperatura e activar o alarme se ela chegar a um determinado valor

CRC (Class Responsibility Card):

Nome da classe	Sensor Temperatura
Responsabilidades	Colaboradores
lerTemperatura()	Alarme
Temperatura	

Mecanismos Comuns

13

- **Notas:** contêm comentários ou restrições ligados a um elemento
- **Estereótipo:** estende o vocabulário do UML permitindo a criação de novos tipos de elementos semelhantes aos já existentes, porém específicos p/ o seu problema. Ex.: <<interface>>
- **Tagged value:** é uma extensão das propriedades do UML e permite adicionar nova informação sobre um elemento. Ex.: {persistent}
- Restrição (Constraint): permite adicionar novas regras. Ex.: {18 < age < 35}
 - ▣ Vamos defini-las usando OCL

Generalização e herança

14

- Classes podem ser organizadas em hierarquia onde uma classe (superclasse) é uma generalização de uma ou mais classes (subclasses)
- Uma subclasse herda os atributos e operações da superclasse e pode adicionar novas propriedades. (E também herda as dependências!)
- Generalização em UML é implementada como herança em OOP

Herança: princípio da substituição

15

- In object-oriented programming, the Liskov substitution principle is a particular definition of subtype that was introduced by Barbara Liskov and Jeannette Wing in a 1993 paper entitled Family Values: A Behavioral Notion of Subtyping [1]. (It is not the only definition; see datatype.)
- The principle was formulated succinctly [2] as follows:
 - ▣ Let $q(x)$ be a property provable about objects x of type T . Then $q(y)$ should be true for objects y of type S where S is a subtype of T .
- Thus, Liskov and Wing's notion of "subtype" is based on the notion of substitutability; that is, if S is a subtype of T , then objects of type T in a program may be replaced with objects of type S without altering any of the desirable properties of that program (e.g., correctness).

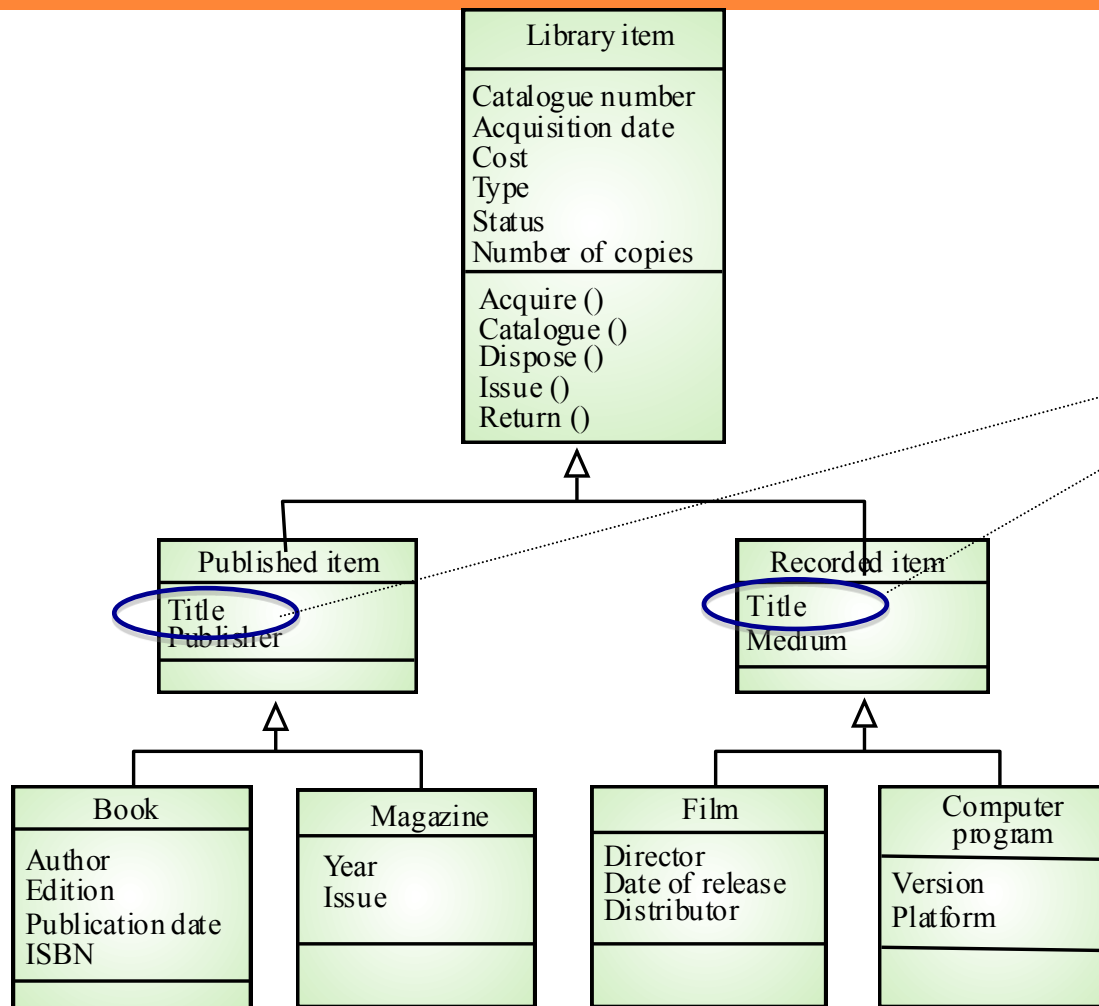
Generalização e herança

16

- Vantagens da herança
 - ▣ É um mecanismo de abstração p/ classificar entidades
 - ▣ É um mecanismo de reutilização
- Problemas com a herança
 - ▣ Classes só são compreendidas completamente com as superclasses
 - ▣ Em casos onde a eficiência é necessária, pode ter que se simplificar o grafo de herança criado na análise.

Hierarquia de classes numa biblioteca

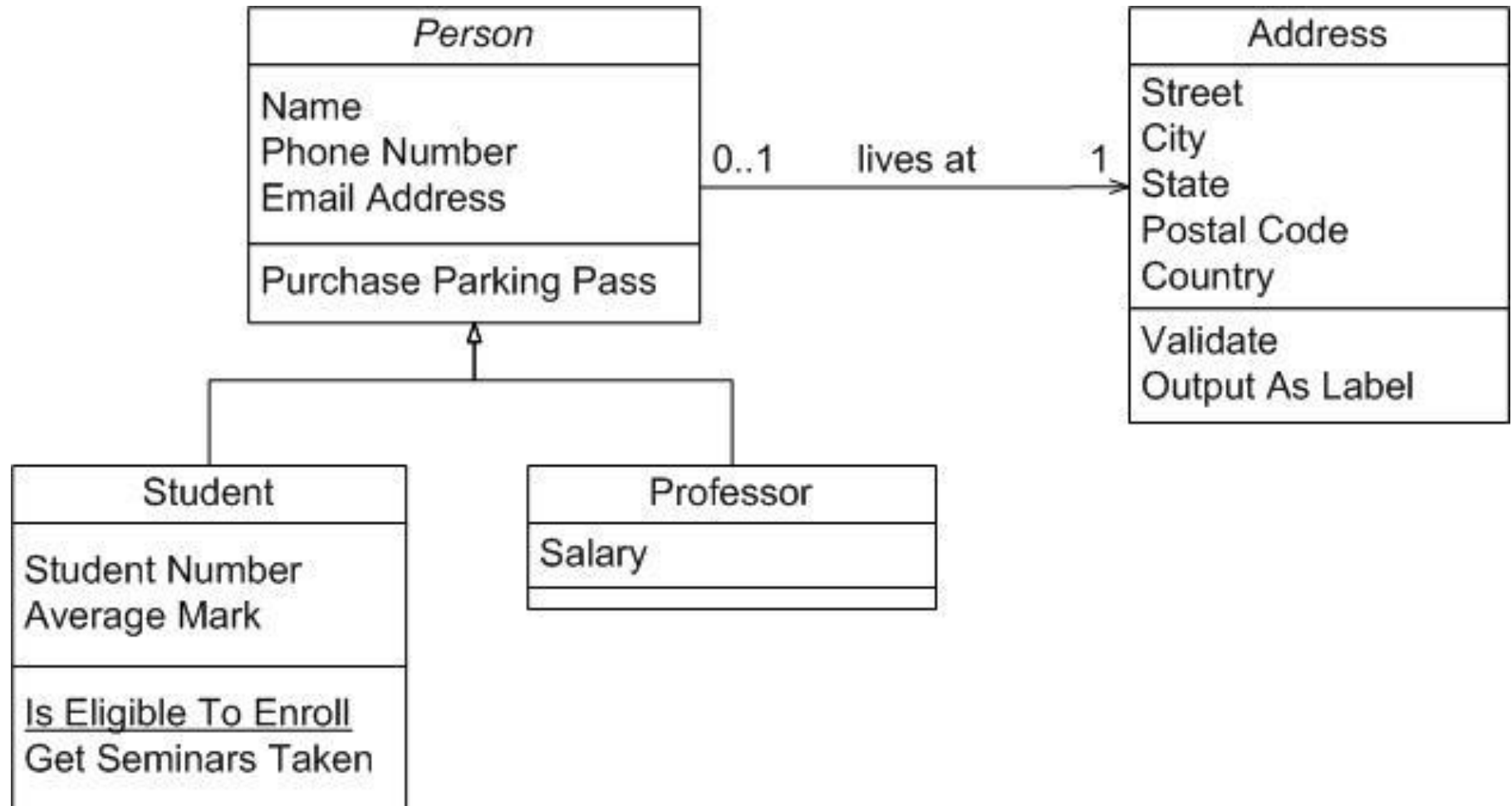
17



Comentários?
Que representa?

Exemplo

18



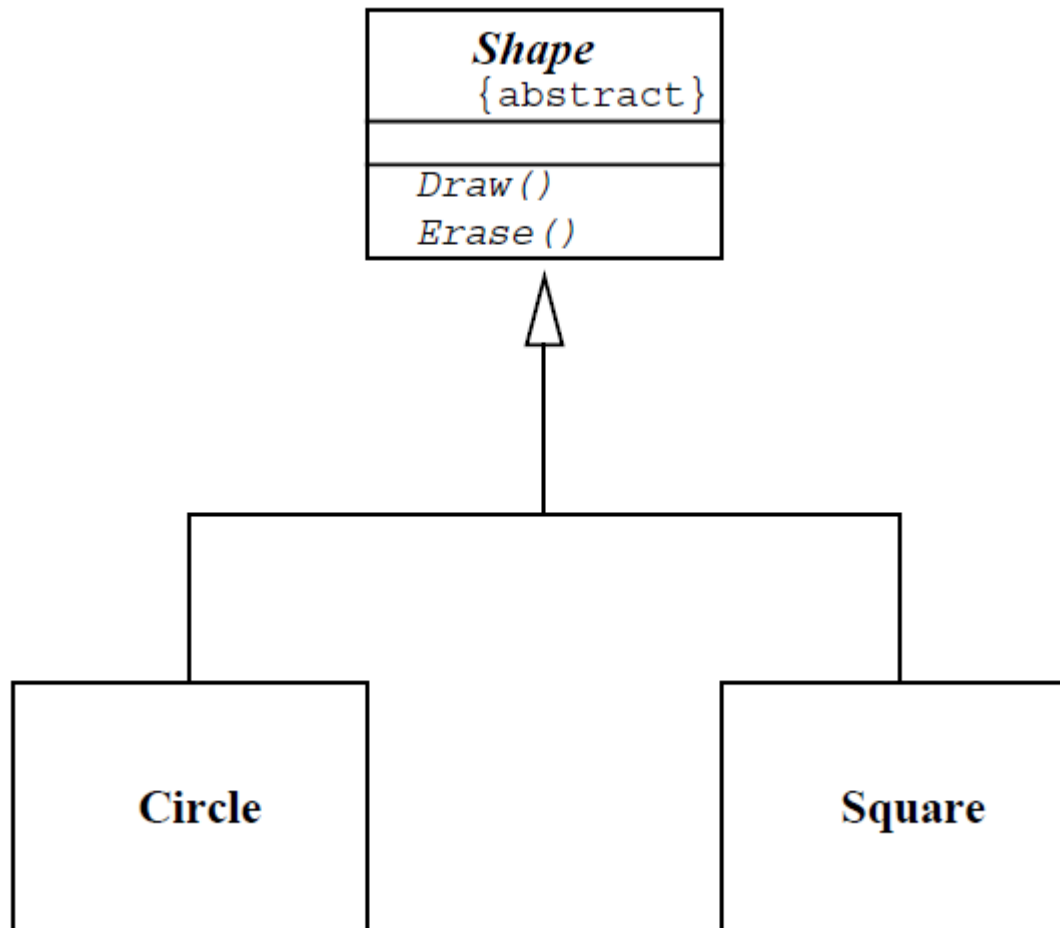
Classes abstractas

19

- Quando a superclasse foi criada para factorizar propriedades comuns a um conjunto de classes de um mesmo tipo, mas que contém uma ou mais operações abstractas (sem implementação oferecida)
- Classes abstractas só são usadas em hierarquias de herança
- São representadas pelo seu nome em itálico, ou então com o estereótipo <<abstract>>, ou o tagged value {abstract}
- As classes abstractas não podem ser instanciadas

Exemplo de classe abstracta

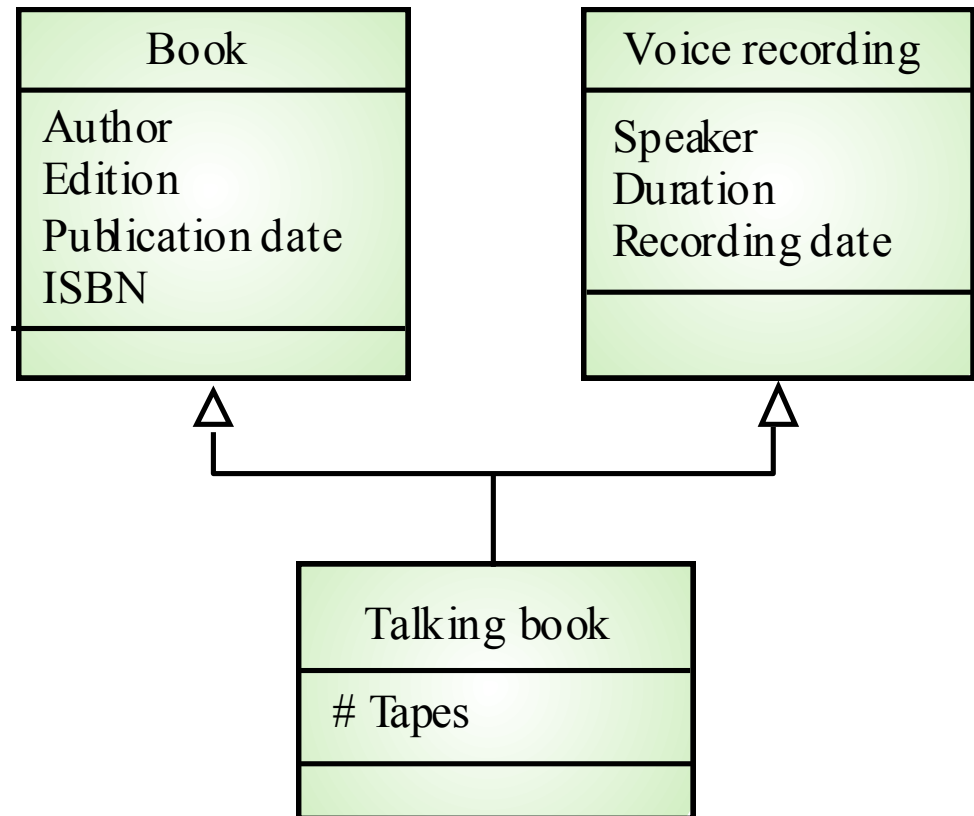
20



Herança múltipla

21

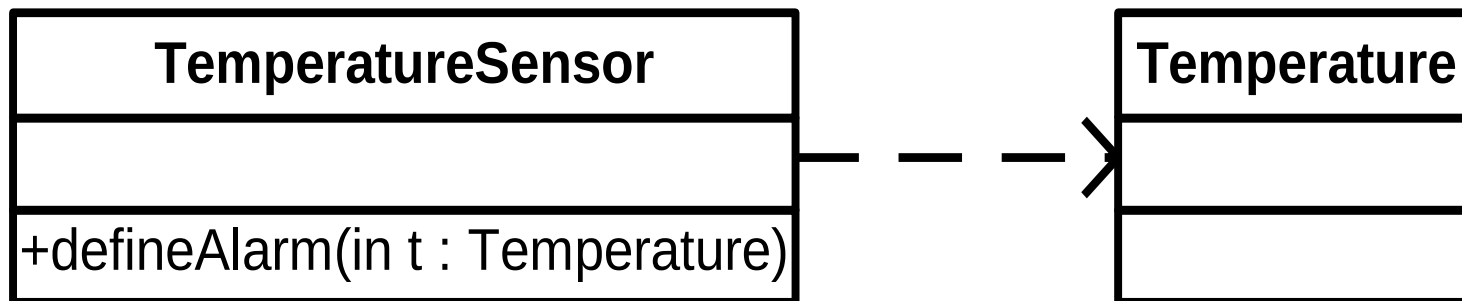
- Uma classe pode herdar atributos e operações de várias superclasses
- Pode levar a conflitos quando atributos e operações de superclasses diferentes tem o mesmo nome e semânticas diferentes
- Torna a hierarquia de classes mais complexa



Dependência

22

- É um relacionamento que determina que uma mudança na especificação de uma classe pode afectar uma outra classe, mas não necessariamente o contrário



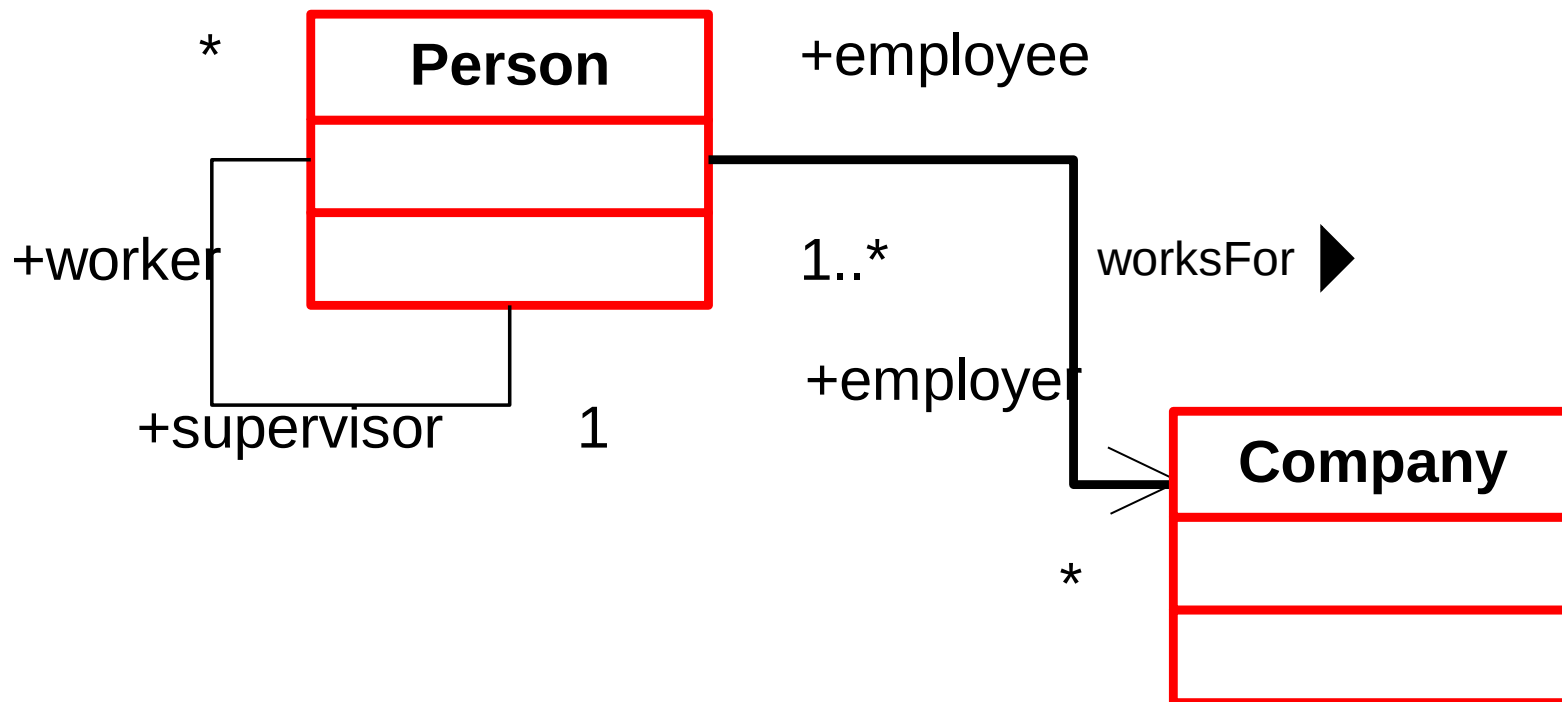
Associações em UML

23

- Objectos de uma classe estão conectados a objectos de outras classes
- Devem ter um nome
- **Papel** (*role*): quando uma classe participa numa associação ela desempenha um papel específico
- **Multiplicidade**: especificação do nº de elementos que um conjunto pode assumir.
 - ▣ Ex.: 1..*, *, 0..1, 3..9, 3..*, etc.
- **Navegação**: mostra como a partir de uma instância de uma classe se pode aceder a uma ou mais instâncias de outra classe

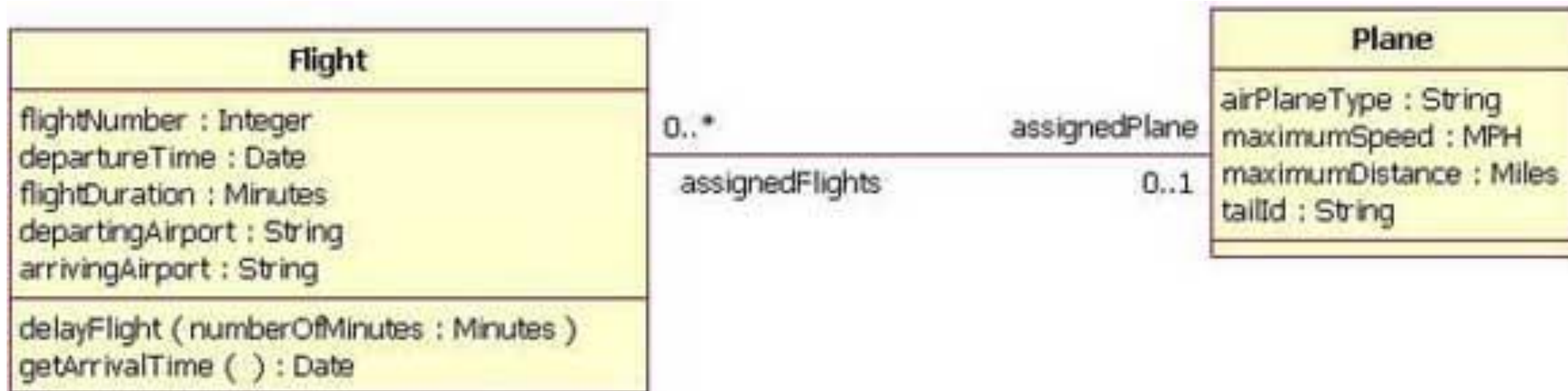
Associações

24



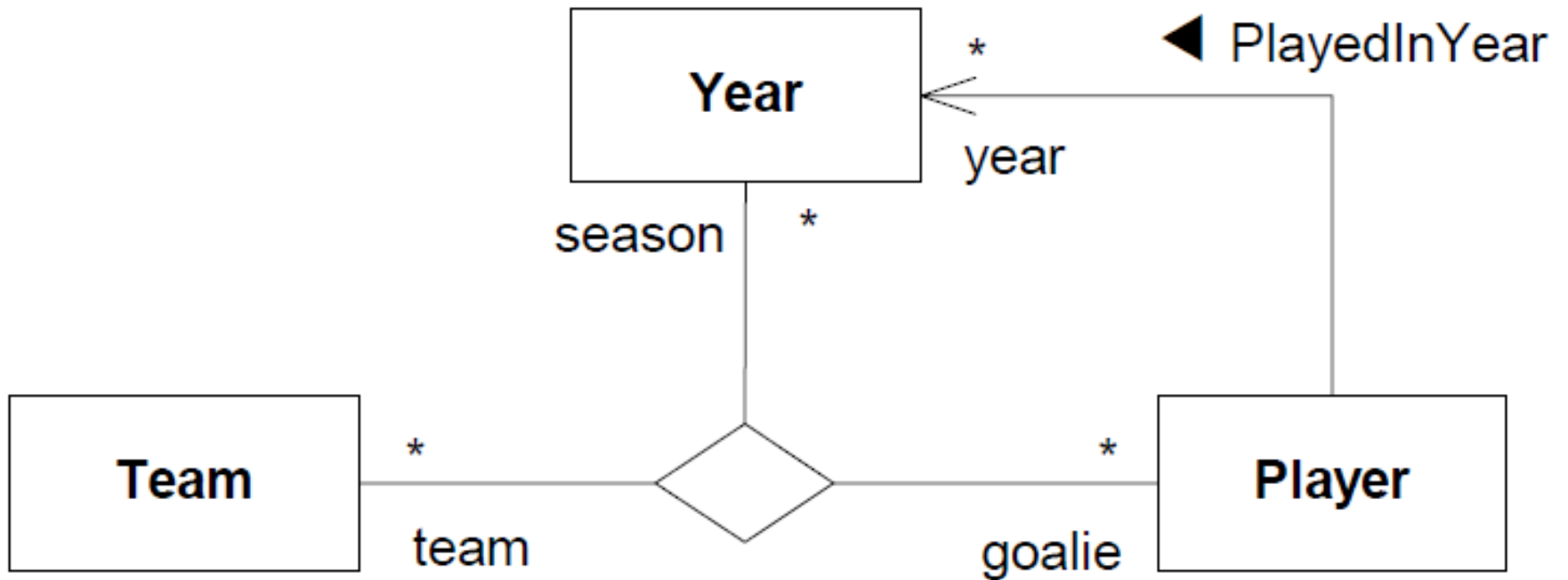
Associação com papéis (*roles*)

25



Associação ternária

26



Associações com restrições

27

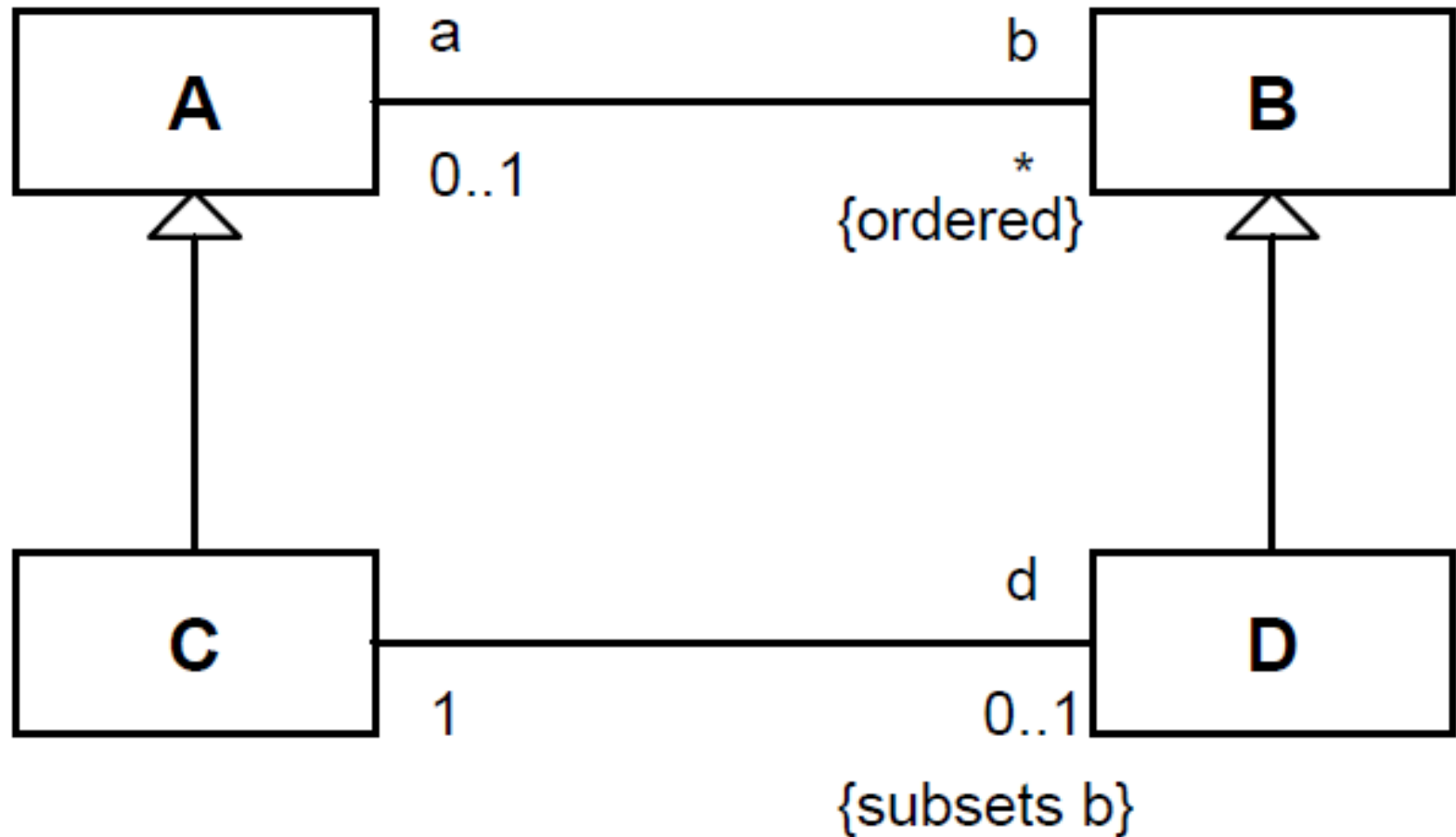
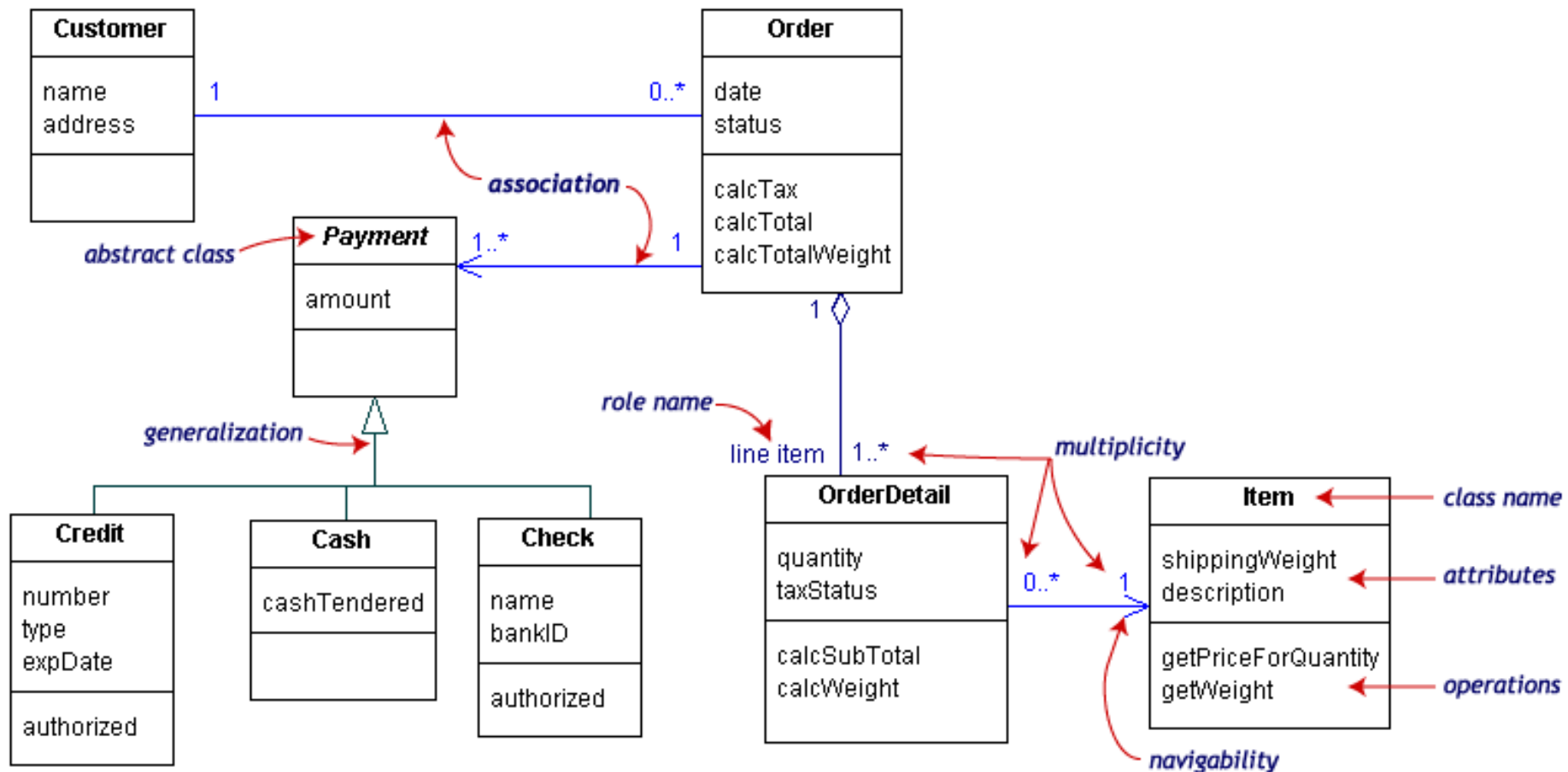


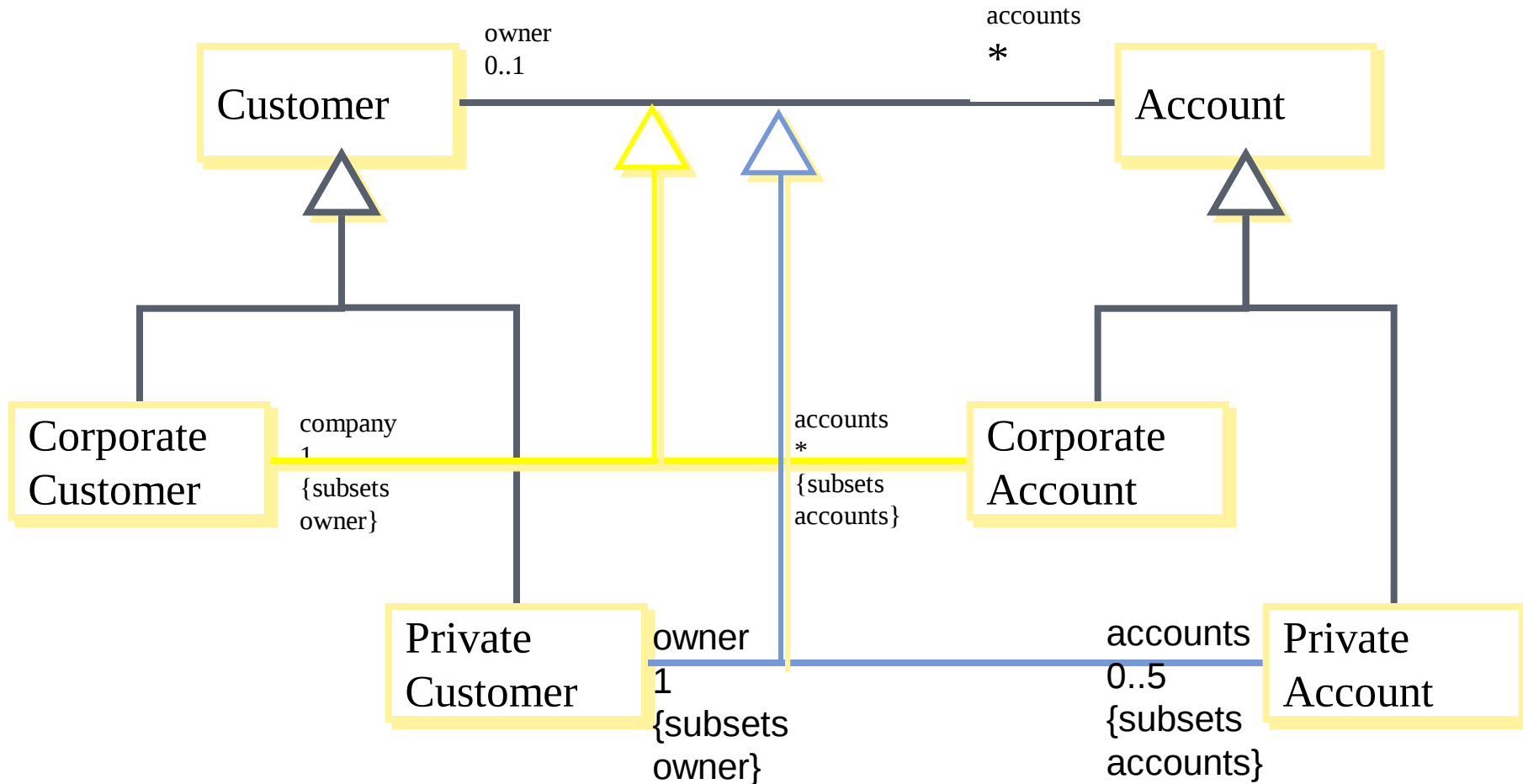
Diagrama de classes: exemplo

28



Especialização de associações

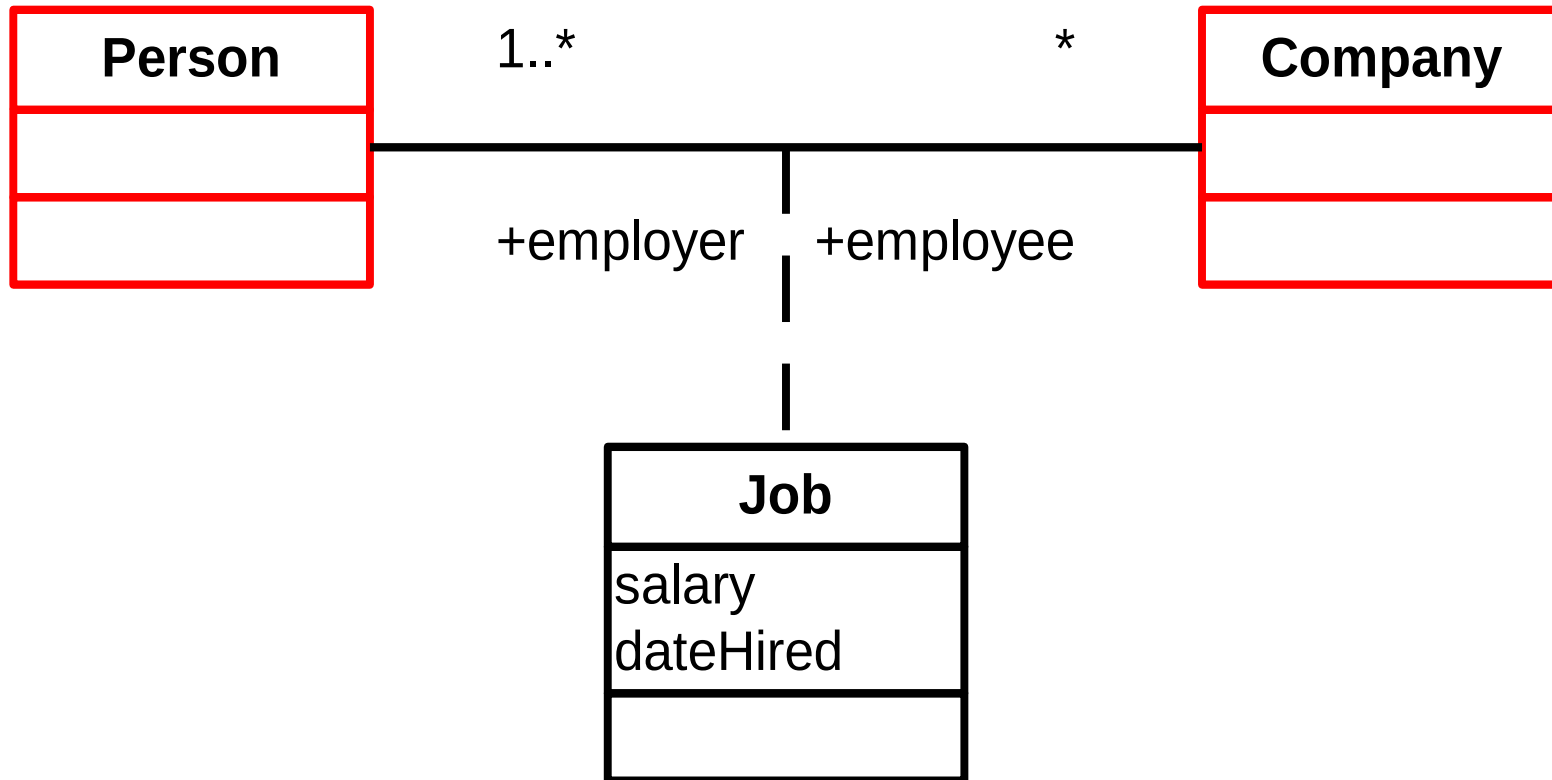
29



Classe-Associação (*association class*)

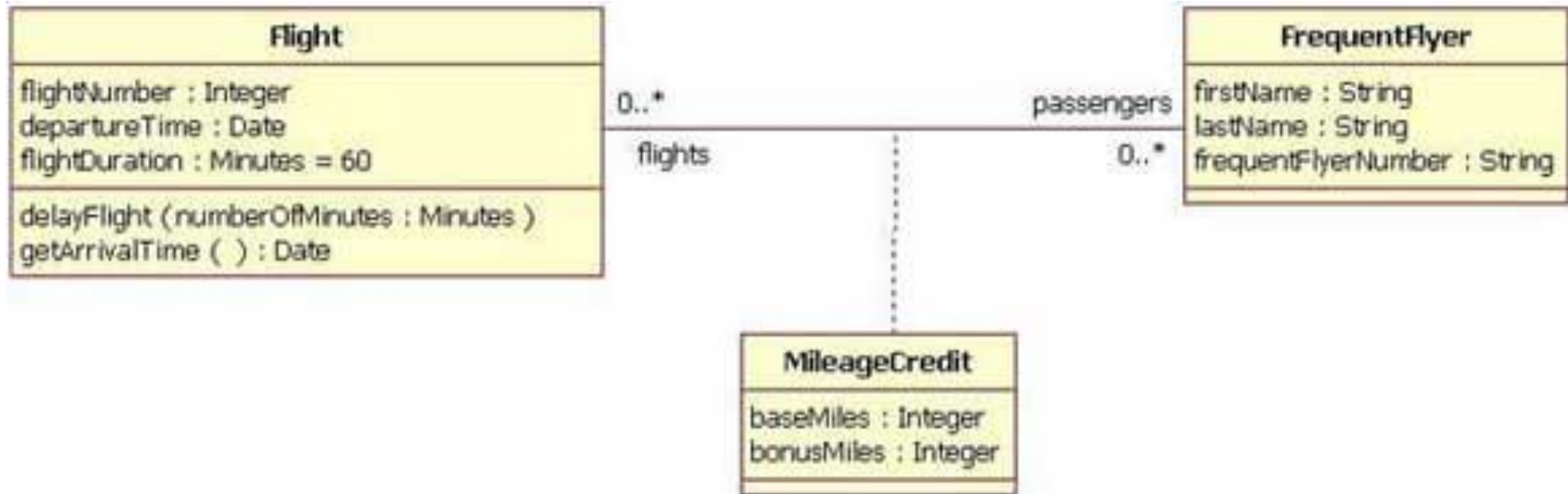
30

- Ocorre quando a associação tem propriedades próprias



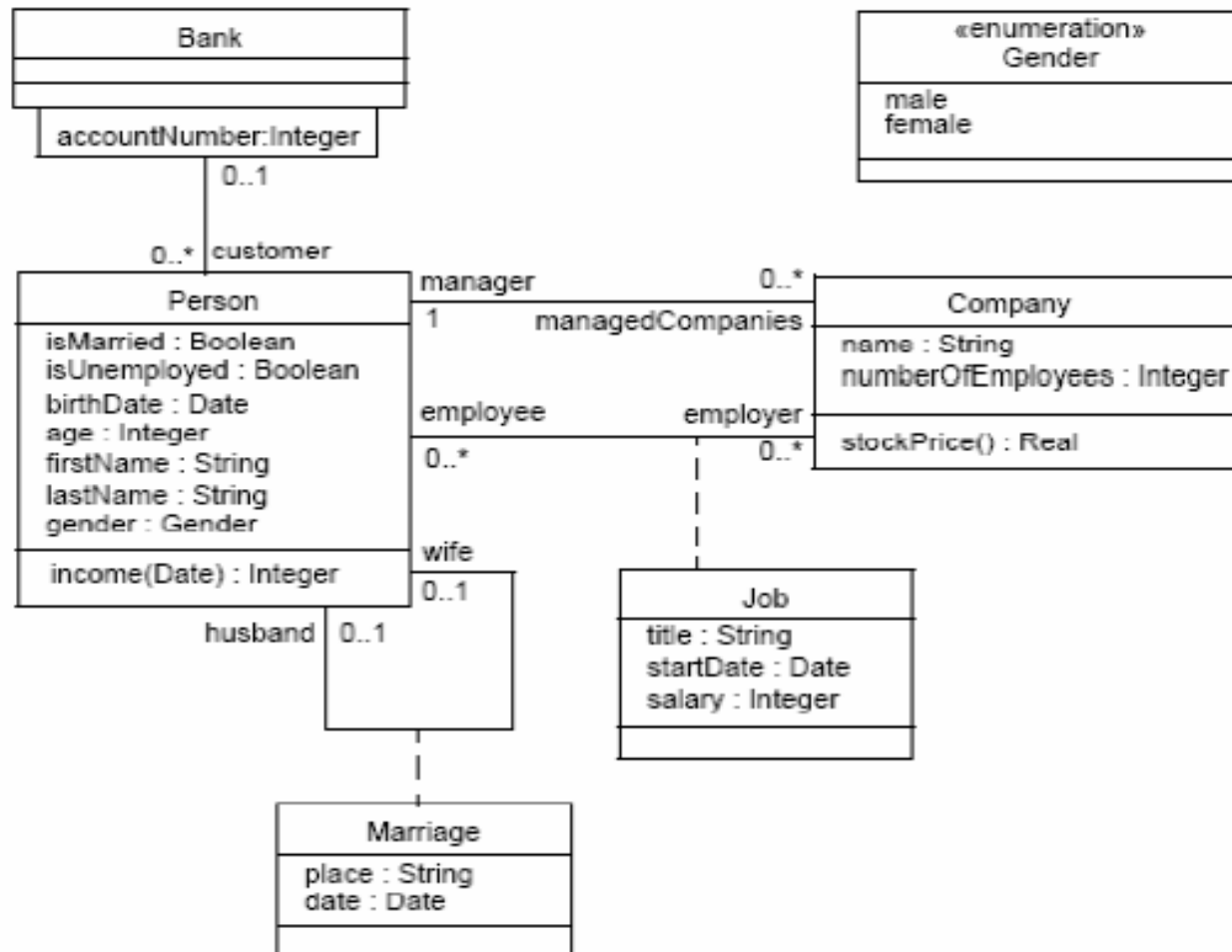
Classe associação (2)

31



Classe associação

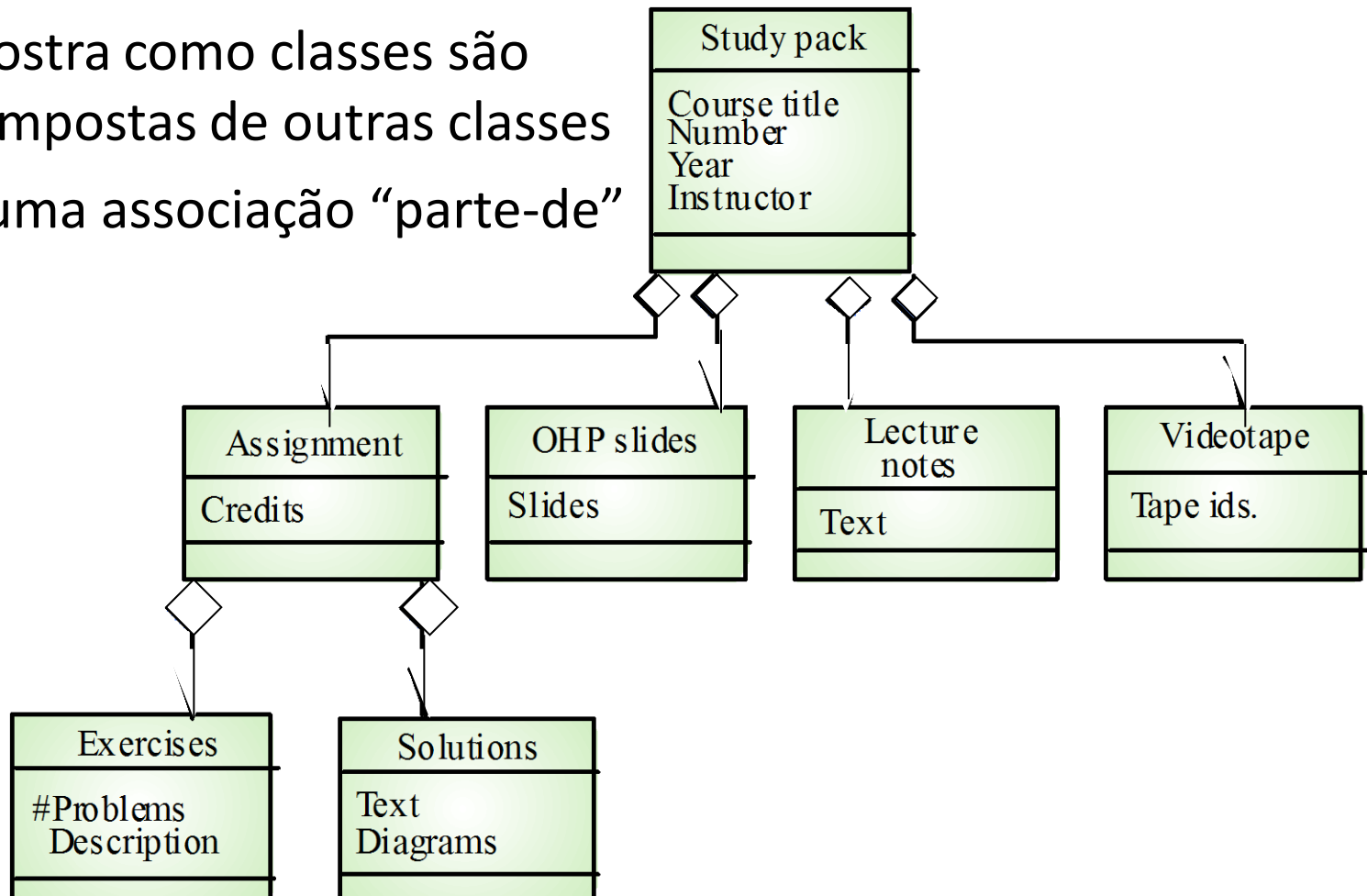
32



Agregação

33

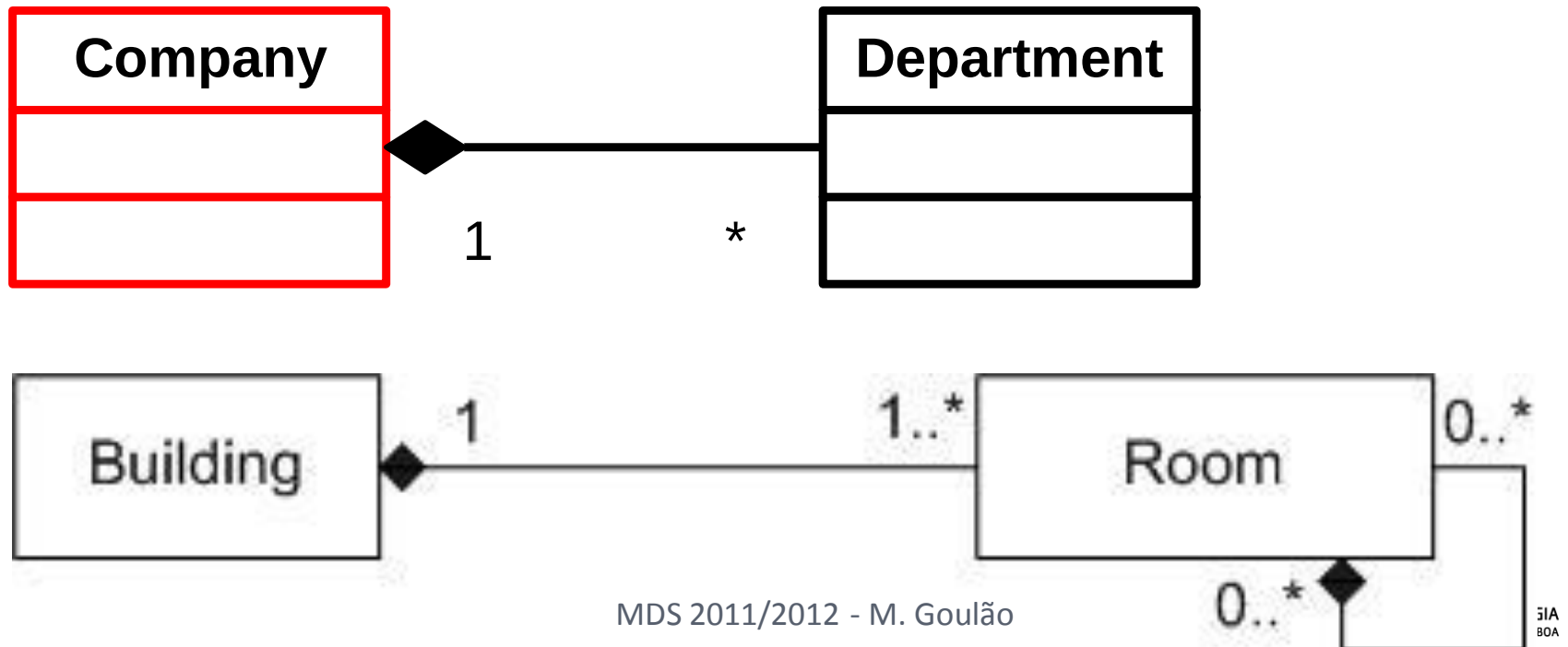
- Mostra como classes são compostas de outras classes
- É uma associação “parte-de”



Composição

34

- É uma forma forte de agregação onde há:
 - ▣ Forte pertença da parte em relação ao todo
 - ▣ As partes não podem existir sem o todo



Agregação e composição

35

