Cadeira de Tecnologias de Informação

Ano lectivo 2010/11

UML – Diagramas de Classes

Tópicos

1. Crise de Software
2. Metodologias Orientadas por Objectos (OO)
3. *Unified Modeling Language* - UML
4. Objectos e Classes
5. Diagramas de Classes
6. Relações entre Classes
7. Como identificar Classes

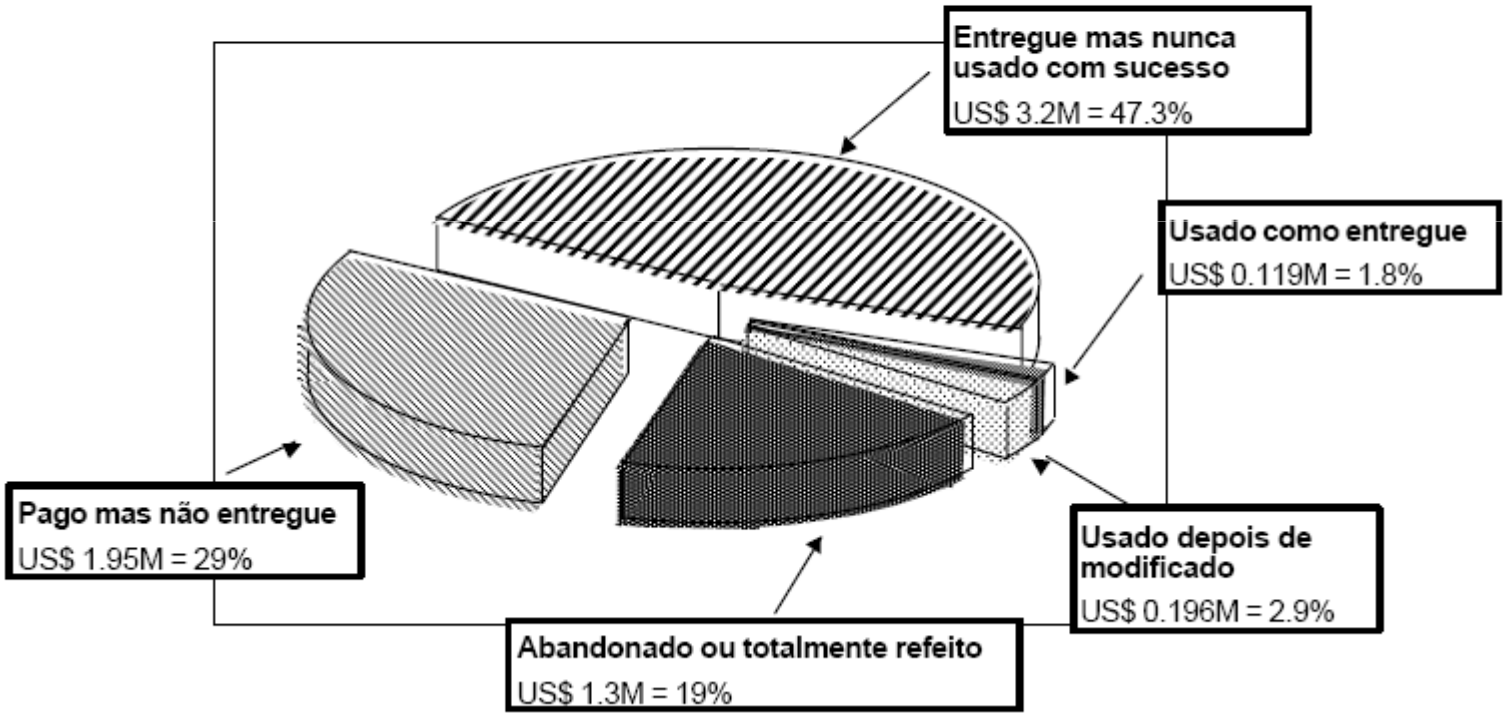
Crise do Software

- Nos finais da década de 60 começou a tomar-se consciência daquilo a que se chamou "crise do software"

- Alguns estudos demonstraram que o software:
  - » raramente respondia às necessidades do cliente
  - » era pouco fiável
  - » excessivamente caro
  - » de manutenção cara e propensa a erros
  - » (o desenvolvimento) excedia os limites de tempo preestabelecidos
  - » era inflexível, não portável e não reutilizável
  - » pouco eficiente, não fazendo um bom uso dos recursos disponíveis

Crise do Software

Resultado do estudo de nove contratos de software, realizado pelo Gabinete de Contabilidade do Governo dos EUA e apresentado ao Congresso, em 1979:



Crise do Software

O software torna-se cada vez mais complexo e não existem técnicas que permitam gerir essa complexidade. Por ex:

- 2/3 dos projectos ultrapassam as estimativas de custos

  Lederer, A. L. & Jayesh P. (1992). Nine Management Guidelines for Better Cost Estimating. *Communications of the ACM 35* (2), 51-59.

- 64% das funcionalidades incluídas nos projectos raramente ou nunca são utilizadas

  Johnson, J. (2002). Keynote speech at Third International Conference on Extreme Programming.

- Cerca de 35% dos projectos ultrapassam o planeamento em 100%

  Standish Group International, Inc. (1994). *Extreme Chaos*. Visto em 25 de Agosto de 2010, em http://standishgroup.com/sample_research/chaos_1994_2.php

Crise do Software

- Pensava-se que para resolver a crise do software era preciso criar linguagens de programação adequadas ao desenvolvimento de grandes (e de qualquer tipo de) sistemas

MAS...

- São necessárias metodologias, técnicas e ferramentas capazes de ajudar a planear, a desenvolver e a gerir o software

FINDING THE GUILTY!

A maior parte dos projectos de software são mal geridos (P.A.N.I.C. Mode...)

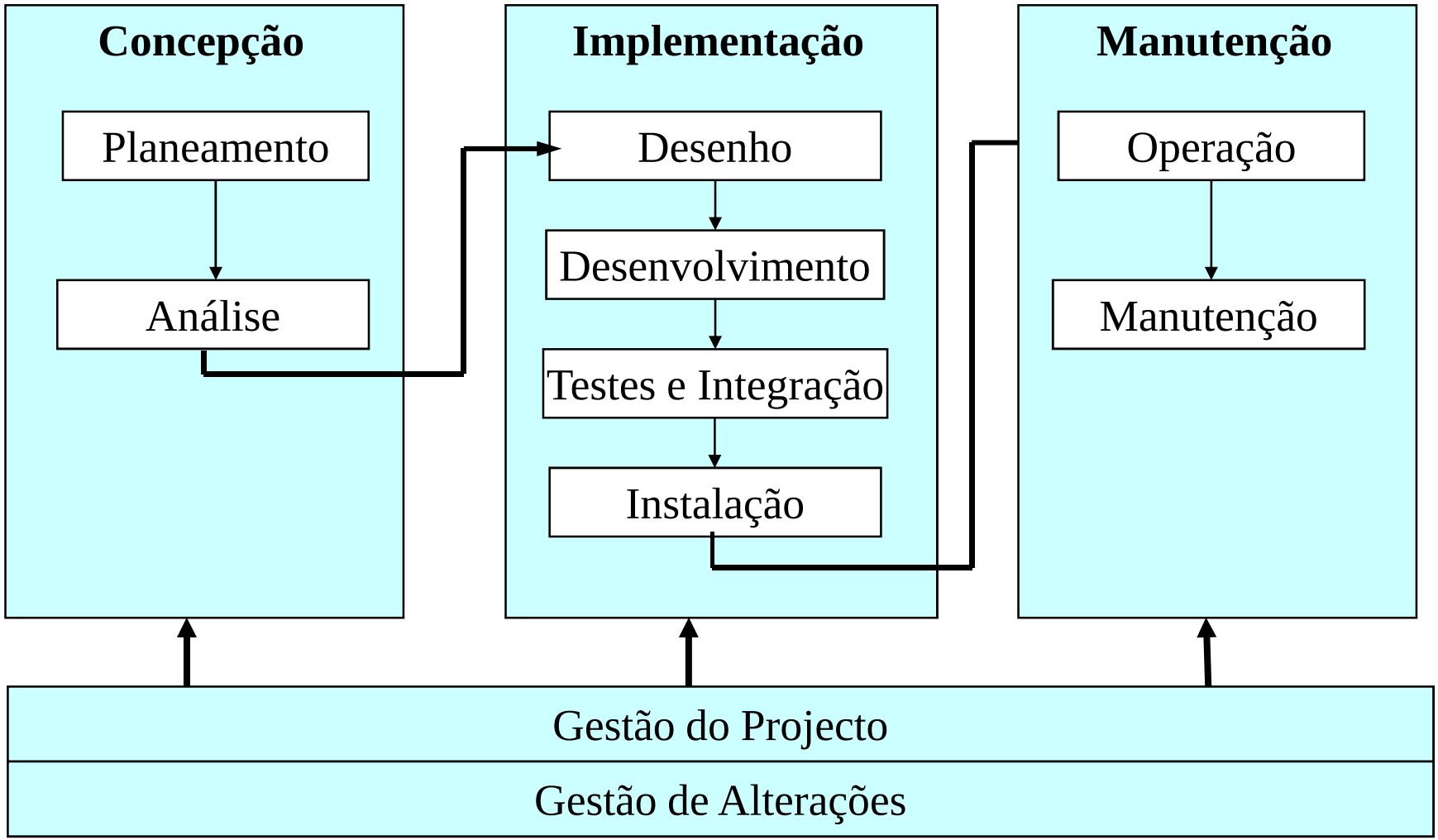
Crise do Software

Para disciplinar o processo de desenvolvimento de software nasceu a Engenharia de Software.

» A **engenharia de software** é a disciplina que define técnicas, metodologias e ferramentas para ajudar a desenvolver e a manter software de qualidade; preocupa-se com todas as etapas desde a definição dos requisitos até a avaliação do produto final.

Fases e Tarefas do Desenvolvimento de S.I.

(1/3) (Silva A., Videira C., 2005)



Fases e Tarefas do Desenvolvimento de S.I.

(2/3) (Silva A., Videira C., 2005)

Concepção

- **Planeamento -** correspondendo a uma identificação geral das necessidades, identificação e selecção de alternativas e definição de plano do trabalho
- **Análise -** inclui a identificação detalhada das funcionalidades do sistema:
  - » **Levantamento de Requisitos** e a respectiva descrição
  - » **Especificação do Sistema** de modo a que os mesmos requisitos possam ser validados pelos utilizadores finais do sistema.

Implementação

- **Desenho,** onde é realizada a definição detalhada da arquitectura global da solução (módulos, tabelas, interface, máquinas, etc.).
- **Desenvolvimento,** tarefa na qual é realizada a programação dos diversos componentes do sistema.

Fases e Tarefas do Desenvolvimento de S.I.

(3/3) (Silva A., Videira C., 2005)

Implementação ....

- **Testes e Integração-** verificações com o objectivo de obter a aceitação do utilizador e migração de dados
- **Instalação,** tarefa onde são executadas as actividades relacionadas com a disponibilização do sistema para os seus utilizadores finais

Manutenção

- **Operação –** funcionamento normal do sistema
- **Manutenção -** corresponde ao tempo de vida útil do SI, durante o qual serão efectuadas todas as alterações posteriores à entrada em funcionamento do produto

Metodologias

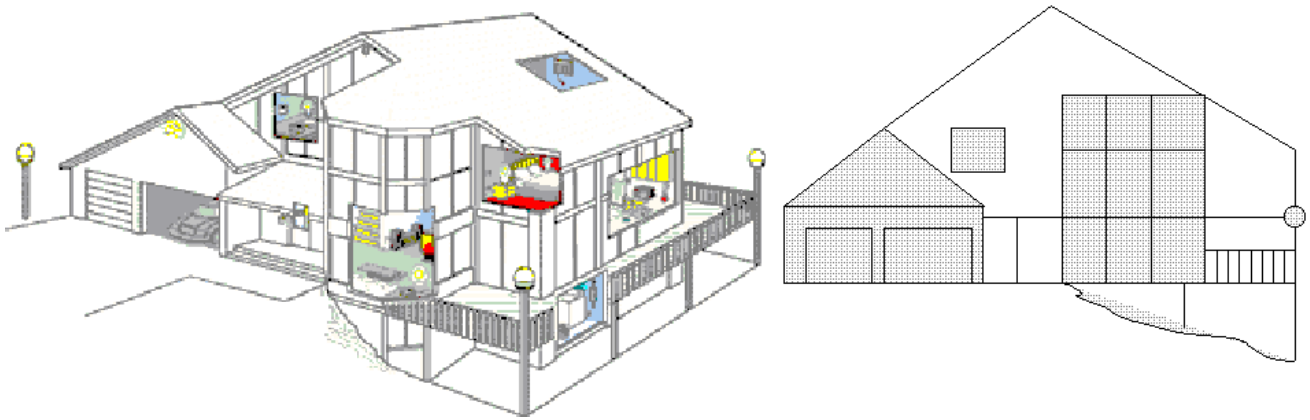
- Durante os anos 70 e 80 apareceram pela primeira vez os conceitos de **ciclo de vida e de metodologia de desenvolvimento de software**, com os seguintes objectivos:
  - » prestar mais atenção ao processo global, e menos à programação
  - » formalizar o processo de identificação de requisitos
  - » Introduzir técnicas baseadas nas melhores práticas no processo de análise e desenho
- Estas metodologias estruturadas (baseadas em processos e dados) propuseram diversos tipos de notações e técnicas de modelação (DFDs, DEAs, etc).

Metodologias - Modelos

Muitas destas metodologias propunham a modelação do sistema, usando modelos para o efeito

Um **modelo** é uma descrição abstracta da estrutura e comportamento de um sistema:

- Os modelos são mais simples de entender que os sistemas que descrevem
- Os modelos podem ajudar-nos a entender e prever o comportamento de um sistema



Os modelos devem ligar os requisitos do negócio com a implementação do software

Metodologias

As técnicas e metodologias inicialmente propostas (estruturadas) apresentaram vários problemas:

- » Dificuldade em lidar adequadamente com a complexidade e tamanho crescente dos sistemas,
- » Difícil integração e reutilização de módulos e componentes do sistema,
- » Qualidade do software baixa com um desempenho inadequado.

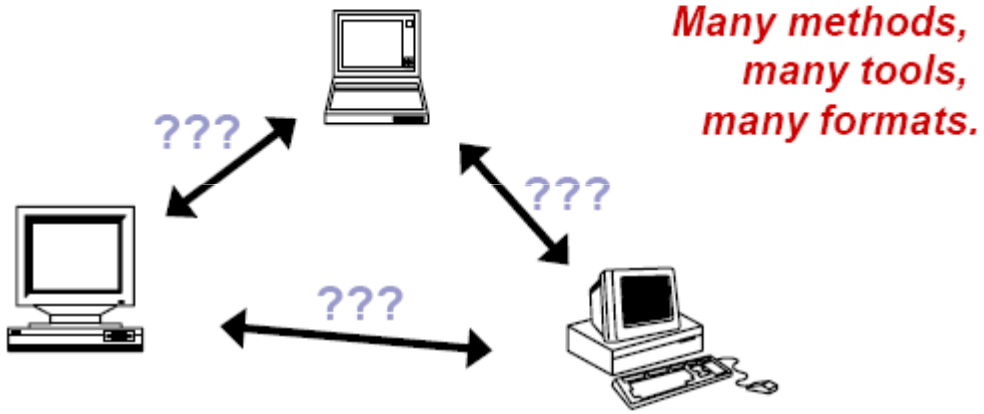
Mais tarde, surgiu o conceito de orientação por objectos (OO), que veio solucionar alguns dos problemas referidos.

Metodologias Orientadas por Objectos

- O paradigma OO baseia-se numa nova forma de analisar o mundo
- Reproduz a forma como o ser humano se apercebe e expressa a realidade que o rodeia
  - » Classifica e subdivide o mundo em diferentes objectos, com base nas diferenças e semelhanças existentes ao nível das características e comportamentos dos mesmos
- As metodologias orientadas por objectos são por isso encaradas como uma das mais recentes propostas para resolver os problemas de desenvolvimento de software
  - » São abordagens mais naturais cujos conceitos básicos são simples e reproduzem o mundo real

Metodologias Orientadas por Objectos

- O elevado número de metodologias disponíveis no mercado fez com que a simbologia usada variasse conforme a metodologia adoptada



- Em meados dos anos noventa a OMG (*Object Management Group*) tomou a iniciativa de criar uma metodologia standard para resolver esta situação

Unified Modeling Language (UML)

- Surge então a *Unified Modeling Language* (UML) na sequência de um esforço de unificação de três das principais linguagens de modelação orientadas por objectos
  - » OMT
  - » BOOch
  - » OOSE

- Em 1997 adquiriu o estatuto de norma

Unified Modeling Language (UML)

- A UML é uma linguagem gráfica para:
  - » Visualizar
  - » Especificar
  - » Construir
  - » Documentar

artefactos de sistemas diversos (software, negócios ou outros), usando o paradigma orientado por objectos

Unified Modeling Language (UML)

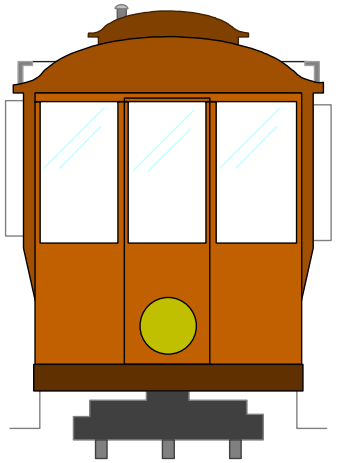
- A versão 2 da UML define 13 tipos de diagramas, divididos em três conjuntos gerais:
  - » **Diagramas de Visão Dinâmica:**
    - – **Diagramas de Estados, Diagramas de Sequência, etc.**
  - » **Diagramas de Visão Estrutural ou Estática:**
    - – **Diagramas de Classes, Diagramas de Componentes, etc.**
  - » **Diagramas de Visão Funcional:**
    - – **Diagramas de Casos de Utilização, Diagramas de Actividades, etc.**
- Cada diagrama apresenta uma perspectiva específica sobre o sistema

Objectos

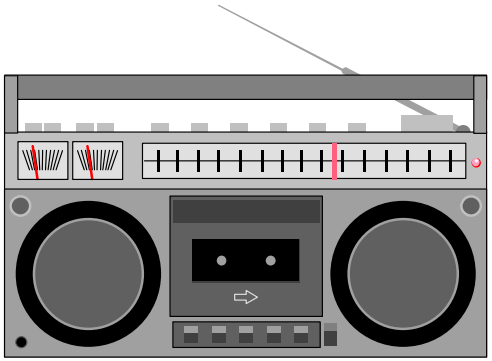
- Um objecto representa uma entidade **física** ou **conceptual** existente no mundo real

- Um objecto tem estado, comportamento e identidade, realizados por:
  - » elementos inertes (*Dados ou Valores)*
  - » elementos dinâmicos (*Métodos)*

Exemplos de Objectos

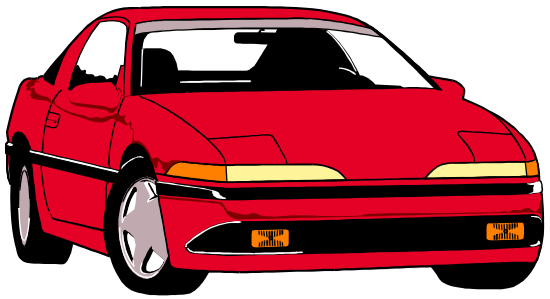


O eléctrico da Praia das Maçãs



O rádio da Cantina

O carro do João



O carro da Maria

Classes

- Uma classe agrega todos os objectos que partilhem as mesmas características, comportamento, relações e semântica

- Pode dizer-se que:
  - » **uma classe é uma abstracção de objectos semelhantes**
  - » **um objecto é uma instância de uma classe**

Exemplos de Classes (1/3)

Pessoa Genérica - Classe

« instâncias de »

Pessoas específicas - Objectos

Cristiano Ronaldo

Barack Obama

Angelina Jolie

Exemplos de Classes (2/3)

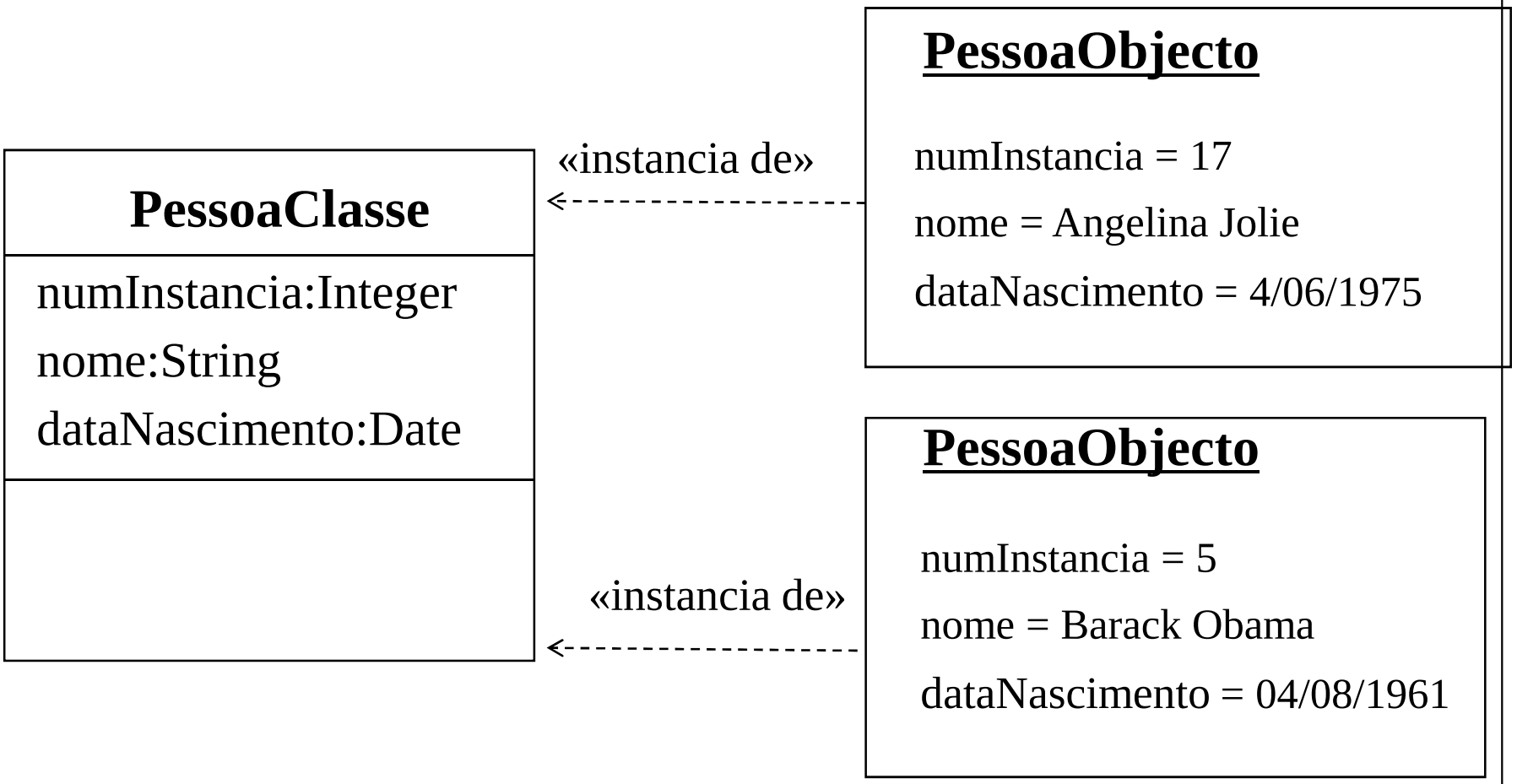
Carros específicos - objectos

«instancia de»

Classe "carro"

Atributos:
tamanho
nº de portas
motor
acessórios
...

Exemplos de Classes (3/3)



Diagramas de Classes

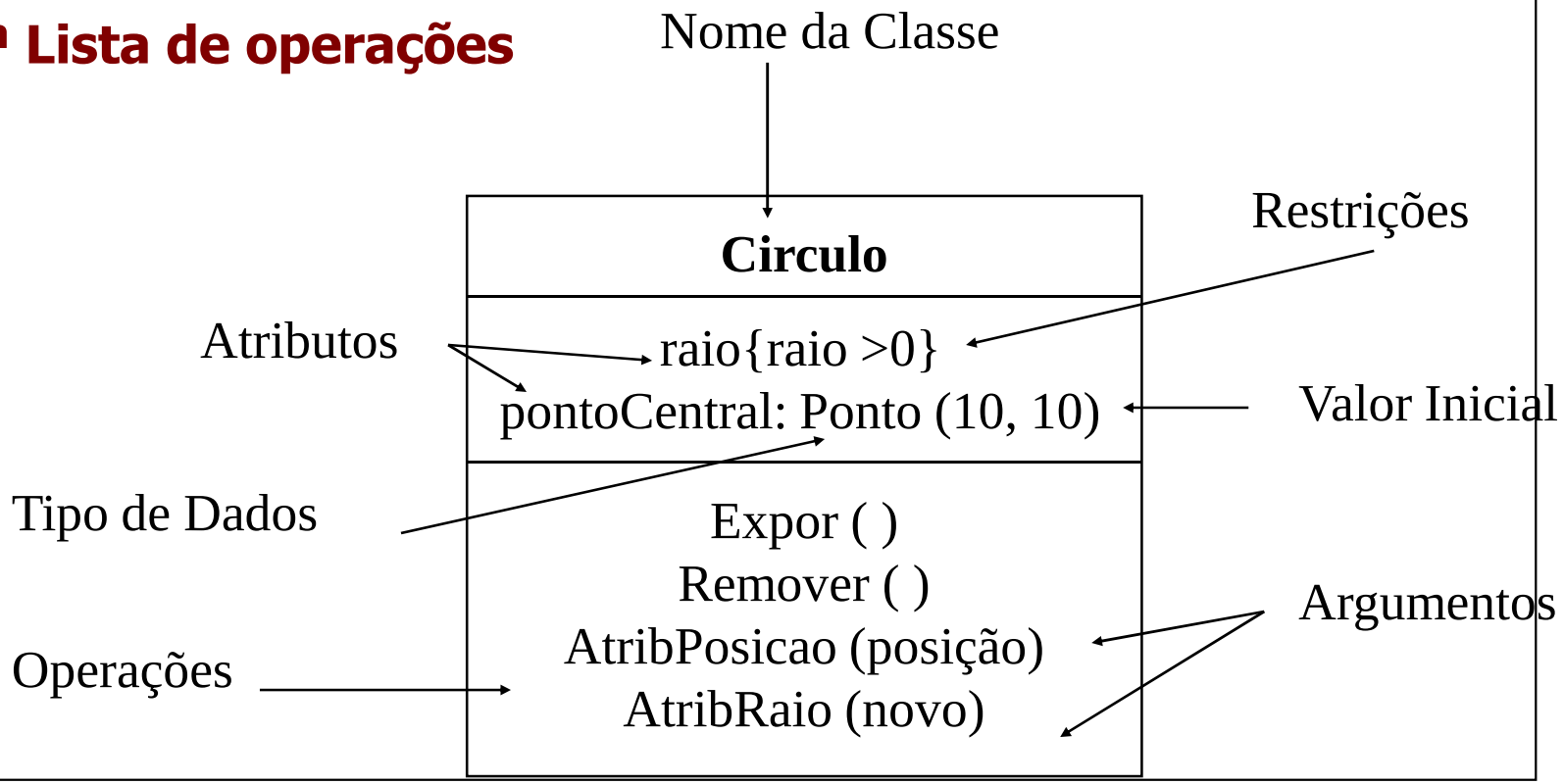
- **São usados para modelar a estrutura de um sistema** e **não descrevem o comportamento do sistema**

- Ilustram um conjunto de:
  - » classes de objectos envolvidos no sistema
  - » relações entre as classes

- Representam-se por um grafo em que:
  - » **os nós representam as classes**
  - » **os arcos representam as relações entre as classes**

Representação da Classe

- Em UML, uma classe é representada por um rectângulo com uma, duas ou três secções:
  - » **1ª Nome da classe**
  - » **2ª Lista de atributos**
  - » **3ª Lista de operações**



Nomes das classes

- Distinguem a identidade da classe
  - » São substantivos retirados do vocabulário do domínio
  - » Devem ser únicos
- Pode ser apresentado na forma simples ou completa

Convenção

Primeira letra de todas as palavras capitalizada

LinhaEncomenda

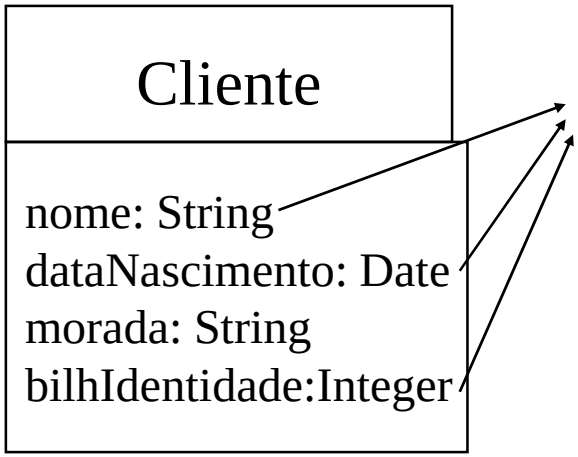
Cliente

Atributos (1/2)

- Um atributo é uma característica que os objectos possuem e que é representada por um valor de dados
- O nome do atributo é obrigatório e tem de ser único no contexto da classe onde é definido

Convenção

Primeira letra de todas as palavras capitalizada, com excepção da primeira

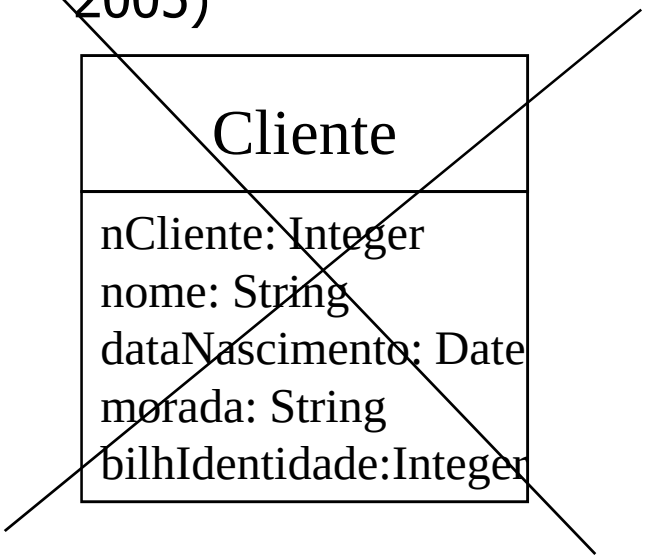


Tipo de atributo:

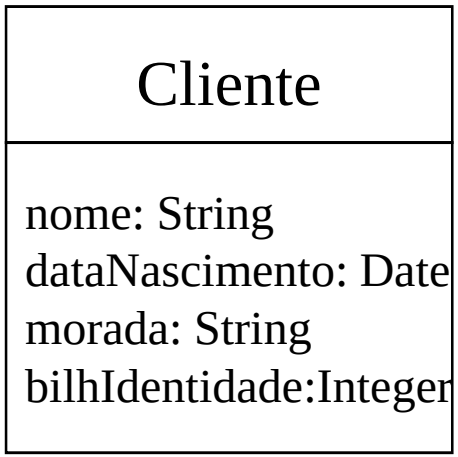
Determina o tipo de informação que pode ser guardada no atributo

Atributos (2/2)

- Não se podem, nem devem, colocar como atributos de uma Classe as chaves "artificiais" (criadas por nós e não correspondendo a atributos existentes no "mundo real"). Ex: codigoOficina, codigoTipoCartão, etc.
- Ex de Atributos existentes no mundo real, que se podem e devem, colocar na Classe: nBI, nContribuinte, nSS,…(Blaha, Rumbaugh, 2005)



Errado

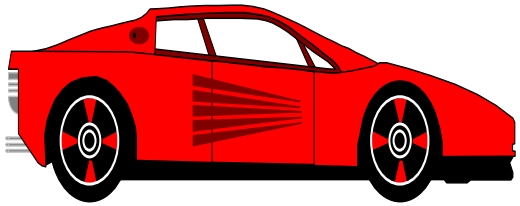


Certo

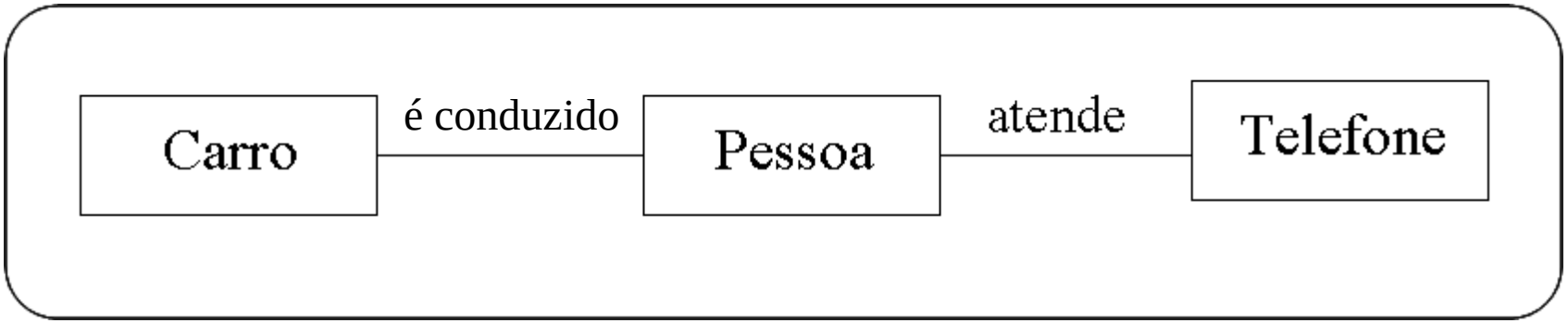
Associações entre as classes

Tal como acontece no mundo real, as classes podem relacionar-se entre si.

Realidade

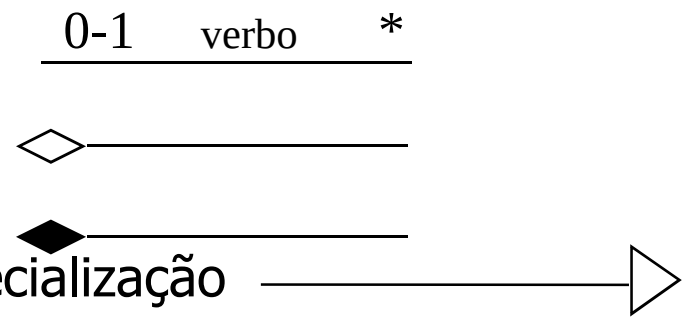


Modelo



Associações entre Classes

- Podemos encontrar os seguintes tipos de relacionamentos entre classes:
  - » Associação
  - » Agregação
  - » Composição
  - » Generalização/Especialização



- Em UML, uma relação estabelece uma ligação entre elementos e é representada graficamente por um determinado tipo de linha

Adornos das Associações (1/2)

- Uma associação é um relacionamento entre diferentes ocorrências de uma ou mais classes; define regras que garantem a integridade desse relacionamento
- As associações representam-se por linhas a cheio complementadas por um conjunto de adornos que especificam diferentes informações:
  - **Nome:** modo de identificar univocamente a associação. Recomenda-se a adopção de um verbo ("emprega") ou de uma expressão verbal ("trabalha para");
  - **Indicador direccional (opcional):** ajuda a esclarecer sentido de leitura da associação. Quando não se coloca, assume-se que a leitura do nome da associação deva ser realizado da esquerda para a direita

Adornos das Associações (2/2)

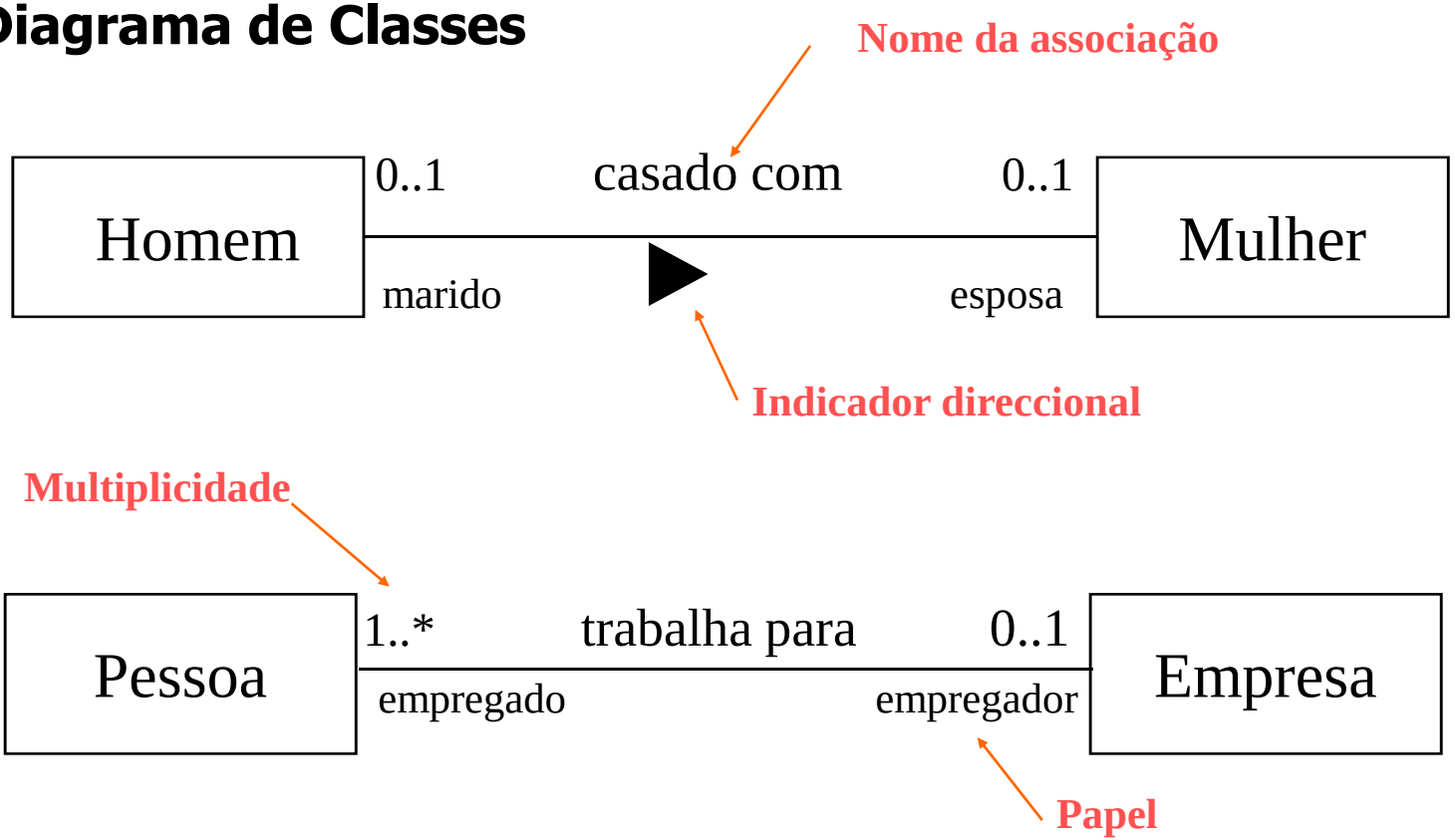
- **Papel de cada participante na associação (opcional):** informa semanticamente como é que um objecto participa na associação

- **Multiplicidade ou cardinalidade:** traduz o número de instâncias de uma classe que se podem relacionar (através da associação) com uma única instância da(s) outra(s) classe(s) participante(s)

- **Navegação:** traduz a forma como a partir de uma instância de uma classe se pode aceder às instâncias da outra classe. Por omissão, uma associação é bidireccional

Representação das Associações

Diagrama de Classes

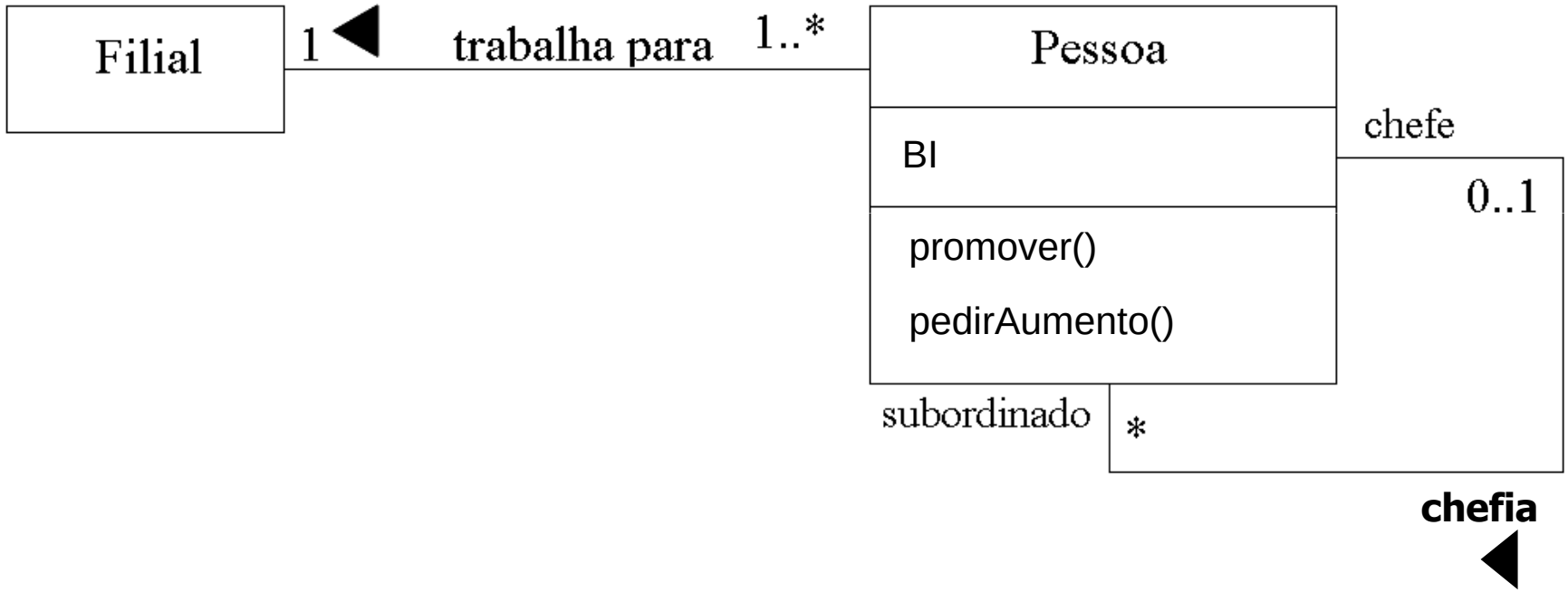


Papéis das Associações

- O extremo de uma associação, no ponto onde liga a uma classe, constitui um dos papéis dessa associação
- O papel é parte da associação e não da classe (i.e., a mesma classe pode ter papéis diferentes em diferentes associações)
- Os papéis são muito importantes, porque ajudam a fazer a representação semântica (que é muito rica) das associações entre classes

Representação dos Papéis das Associações

Os papéis podem ser explicitados ou não



Multiplicidade

- A Multiplicidade é definida em cada extremo da associação por um limite mínimo e um limite máximo:
  - » Limite mínimo:
    - – 0
    - – 1
    - – N (quando é um número conhecido, p.e. 3)
  - » Limite máximo:
    - – 1
    - – M (quando é um número conhecido, p.e. 10)
    - – * (significa "muitos", não é conhecido o limite)
- A representação na associação é feita como um intervalo entre o limite mínimo e o limite máximo.

Multiplicidade das Associações

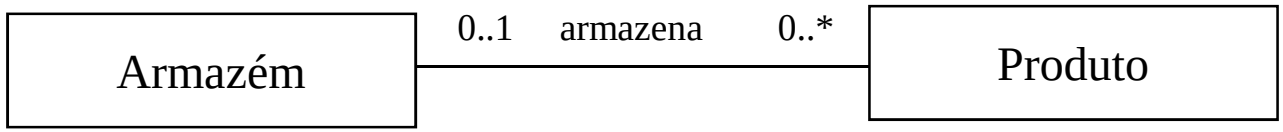
Exemplos de multiplicidade:

- * ou (0..*) – muitos (0 ou mais);
- (1..*) – um ou mais;
- (0..1) – zero ou um;
- 1 ou (1..1) – exactamente um;
- (e.g., 5) – um determinado número;
- (e.g., 2..5) – uma determinada gama; ou mesmo uma multiplicidade mais complexa especificada através de listas (p.e.: 0..3, 5..7, 10..).

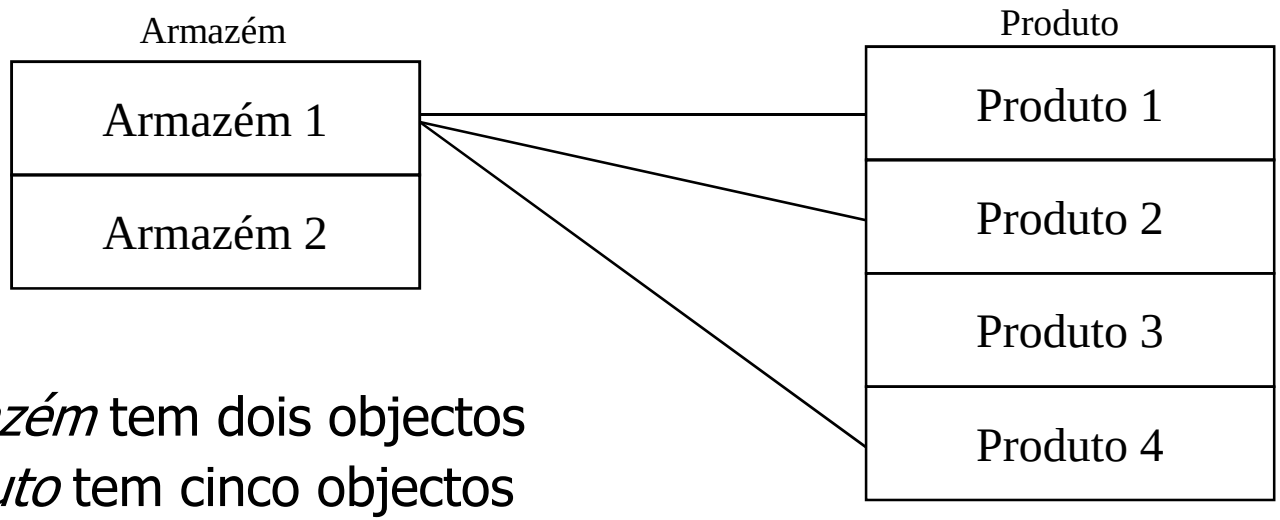
Multiplicidade

Associação é uma relação estrutural onde objectos de uma classe estão ligados a objectos de outra classe

Diag. de Classes



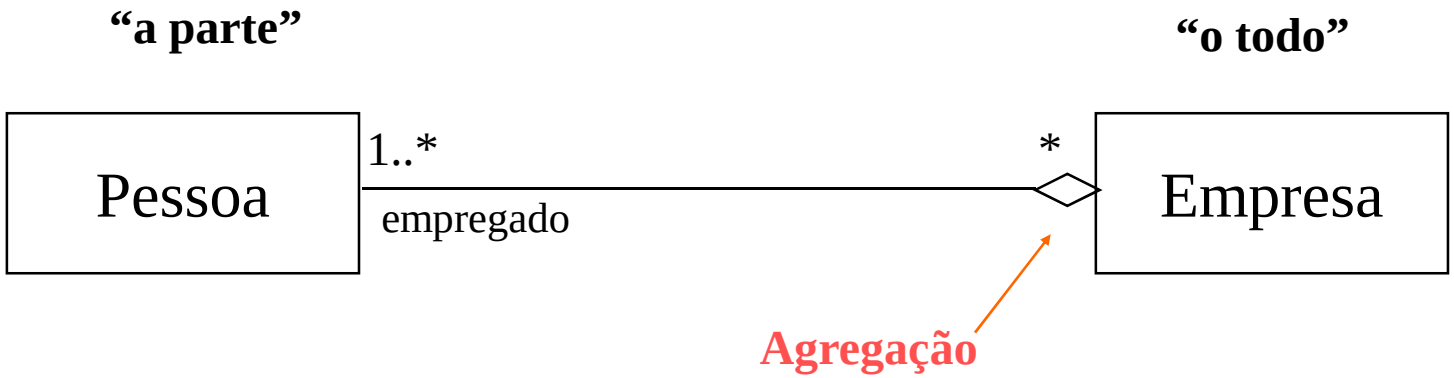
Objectos



- A classe *Armazém* tem dois objectos
- A classe *Produto* tem cinco objectos
- A associação *armazena* está a ligar um objecto de *Armazém* com três objectos de *Produto*

Agregação

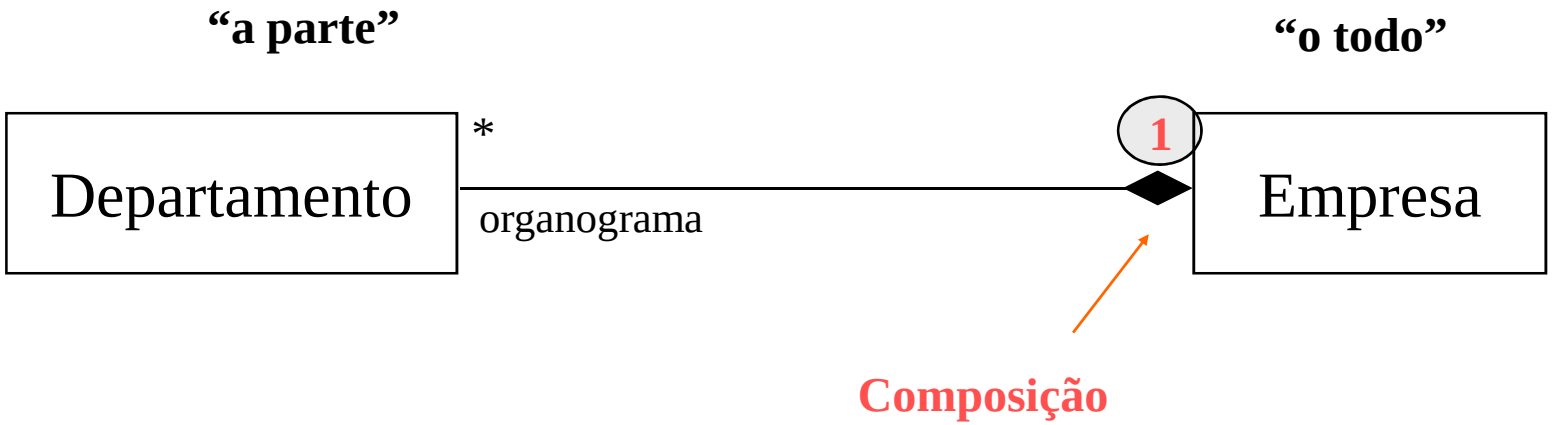
- É uma forma especial de associação que traduz que existe uma relação de "é parte de" ou "tem".
- Representa-se por um losango não preenchido colocado junto à classe que representa o elemento agregador ou "o todo".



Importante: A parte pode permanecer sem o todo!

Composição

- Também designada de agregação forte, é uma variante à agregação simples, em que é adicionada semântica
- Representa-se por um losango preenchido colocado junto à classe que representa o elemento agregador ou "o todo"

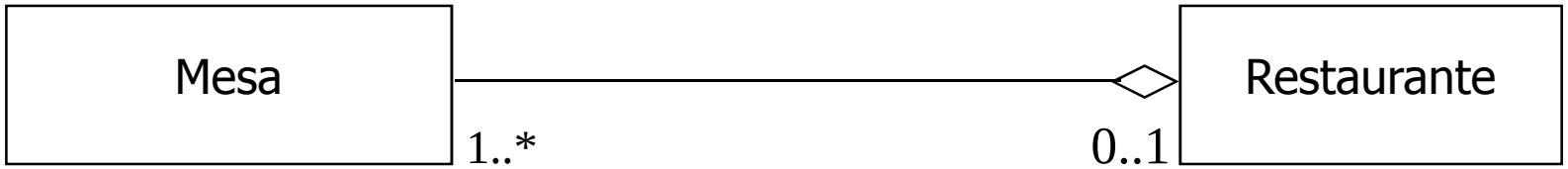


Importante: As partes não podem existir sem o todo!

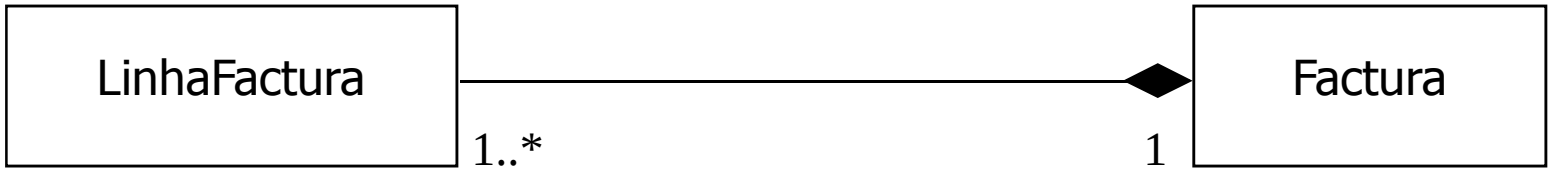
Mais exemplos

Agregação

O todo é composto por partes



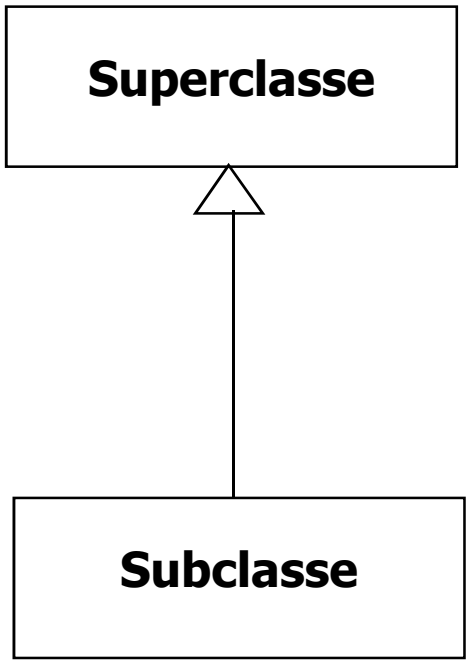
Composição



Uma linha de factura não existe fora do contexto de uma factura

Generalização

Representa um relacionamento entre uma classe (superclasse) e uma ou mais variações dessa classe (as subclasses), na perspectiva de uma relação entre um elemento geral e um elemento mais específico.



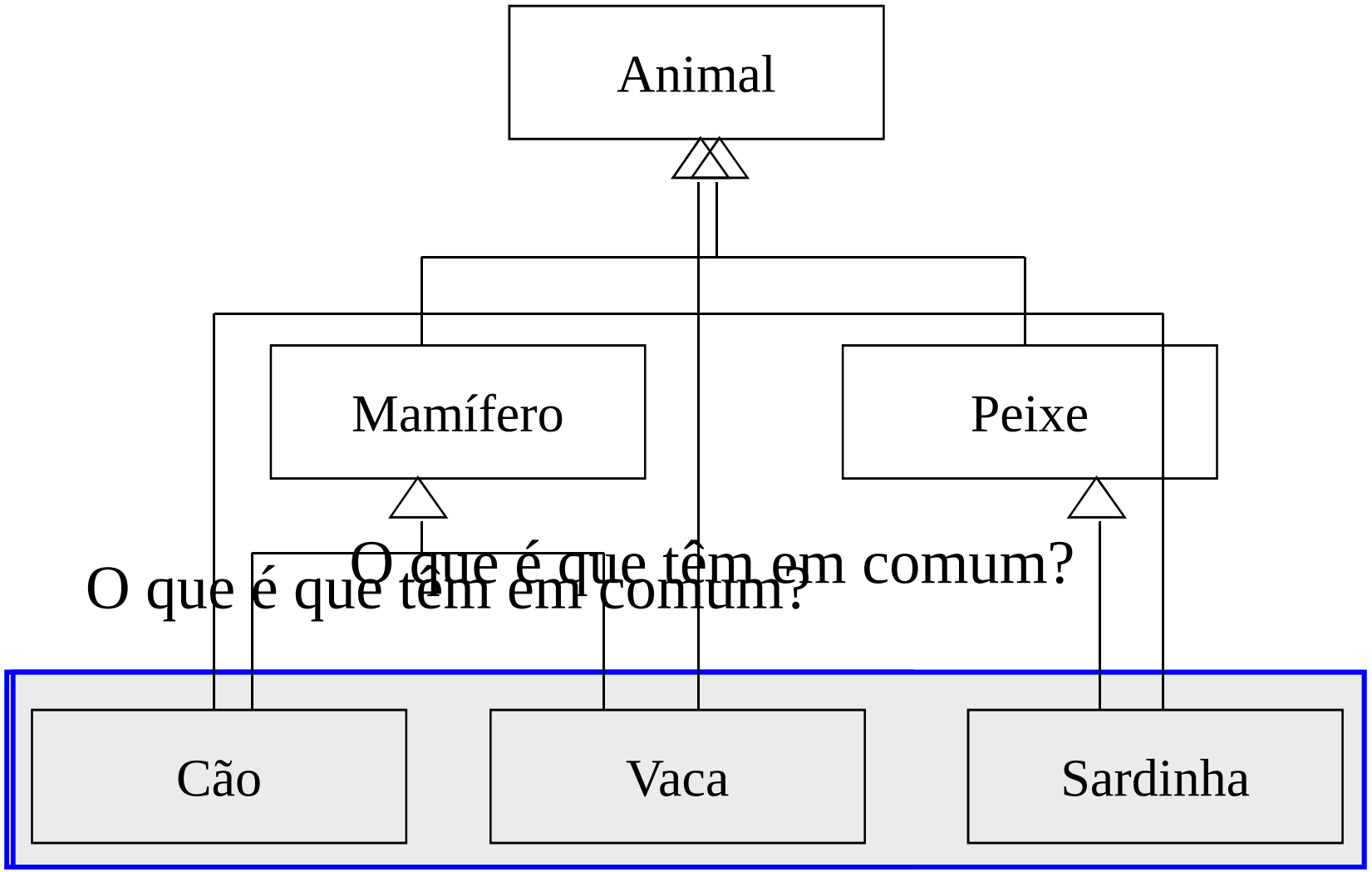
Representa-se em UML por uma linha a cheio com um triângulo a branco num seu extremo

Generalização

- No contexto das classes usam-se generalizações para ilustrar o conceito de herança

- A herança providencia um mecanismo natural de organização:
  - » **Cada subclasse herda o estado (atributos) e o comportamento (operações) da superclasse**
  - » **As subclasses podem adicionar atributos e comportamentos específicos**

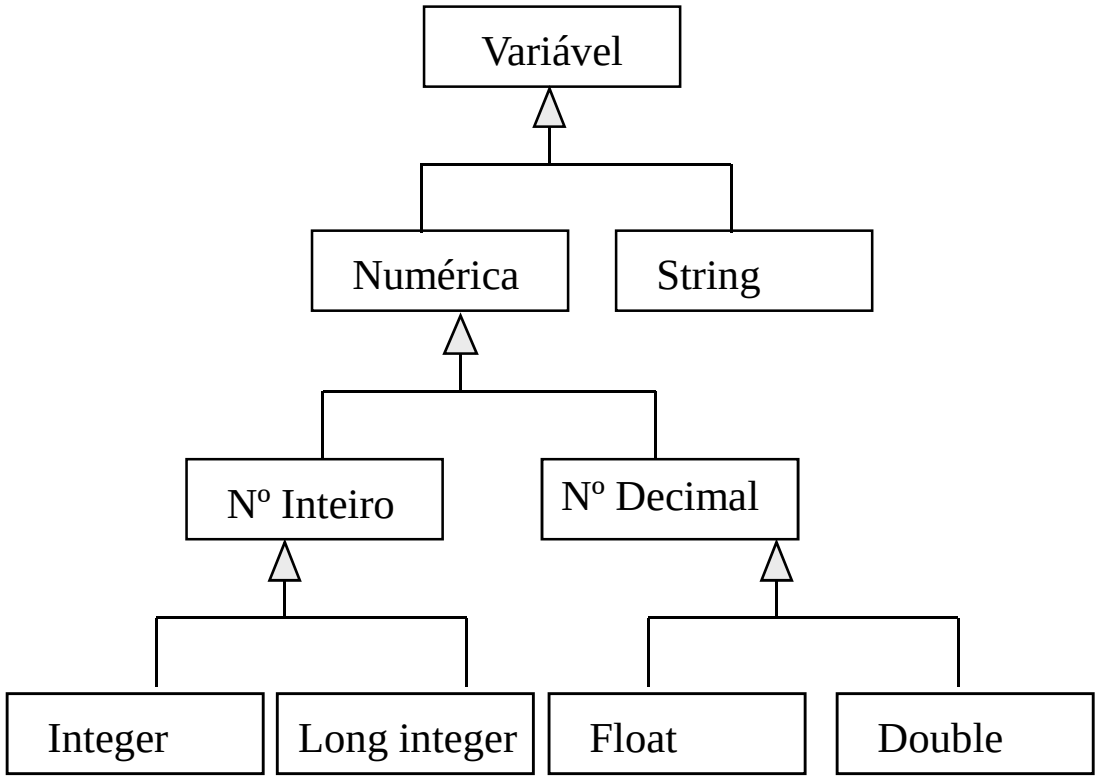
Exemplo de Herança



Exemplo de Herança - Variáveis

super-classes

Generalização

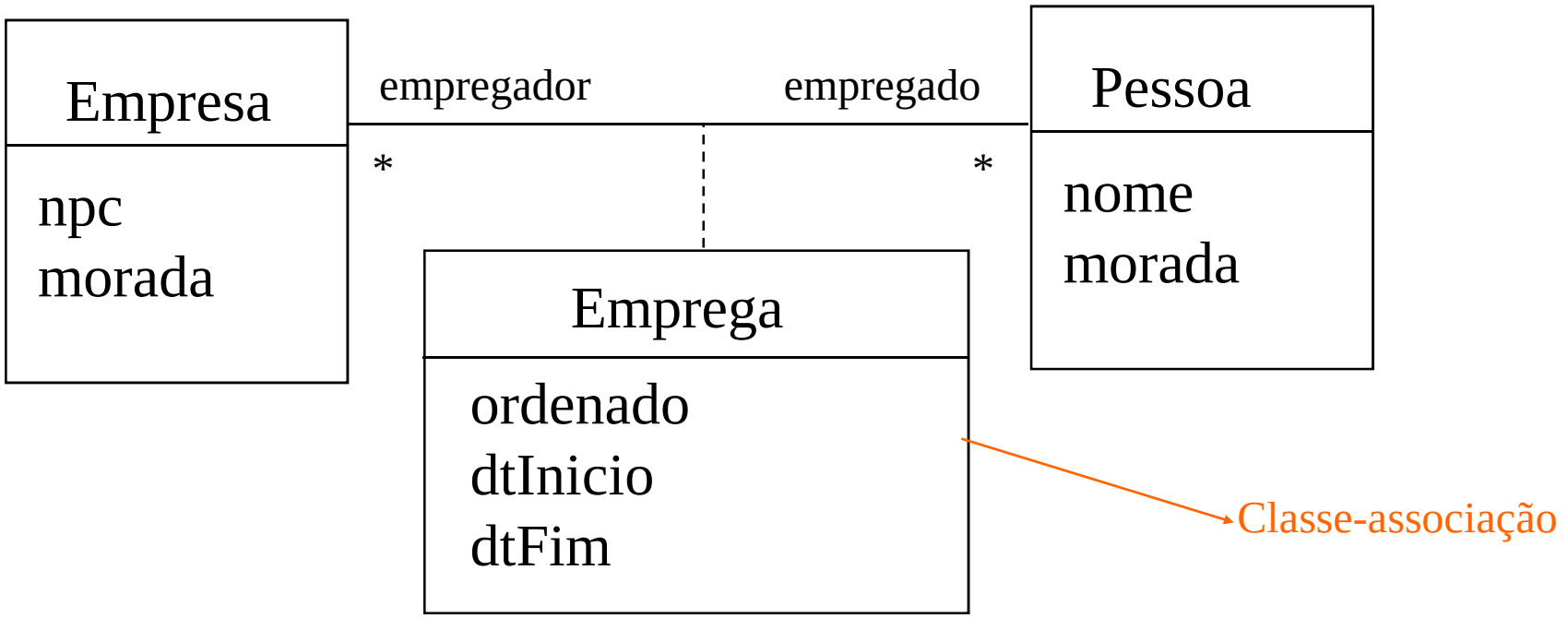


Especialização

sub-classes

Classe-Associação ou Associação Atributiva

Quando uma associação tem os seus próprios atributos deve ser representada como uma classe



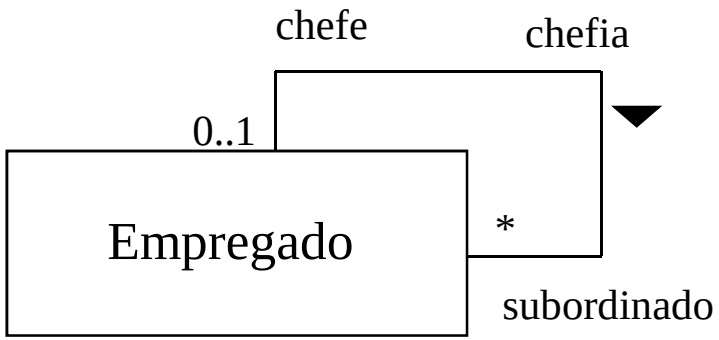
Grau das Associações

- O Grau traduz o número de classes que intervêm numa dada associação

- As associações podem ser
  - » Binárias – associação entre duas classes (caso mais frequente)
  - » Unárias (ou Reflexivas) – associação de uma classe consigo própria
  - » N-árias – associação entre mais de duas classes

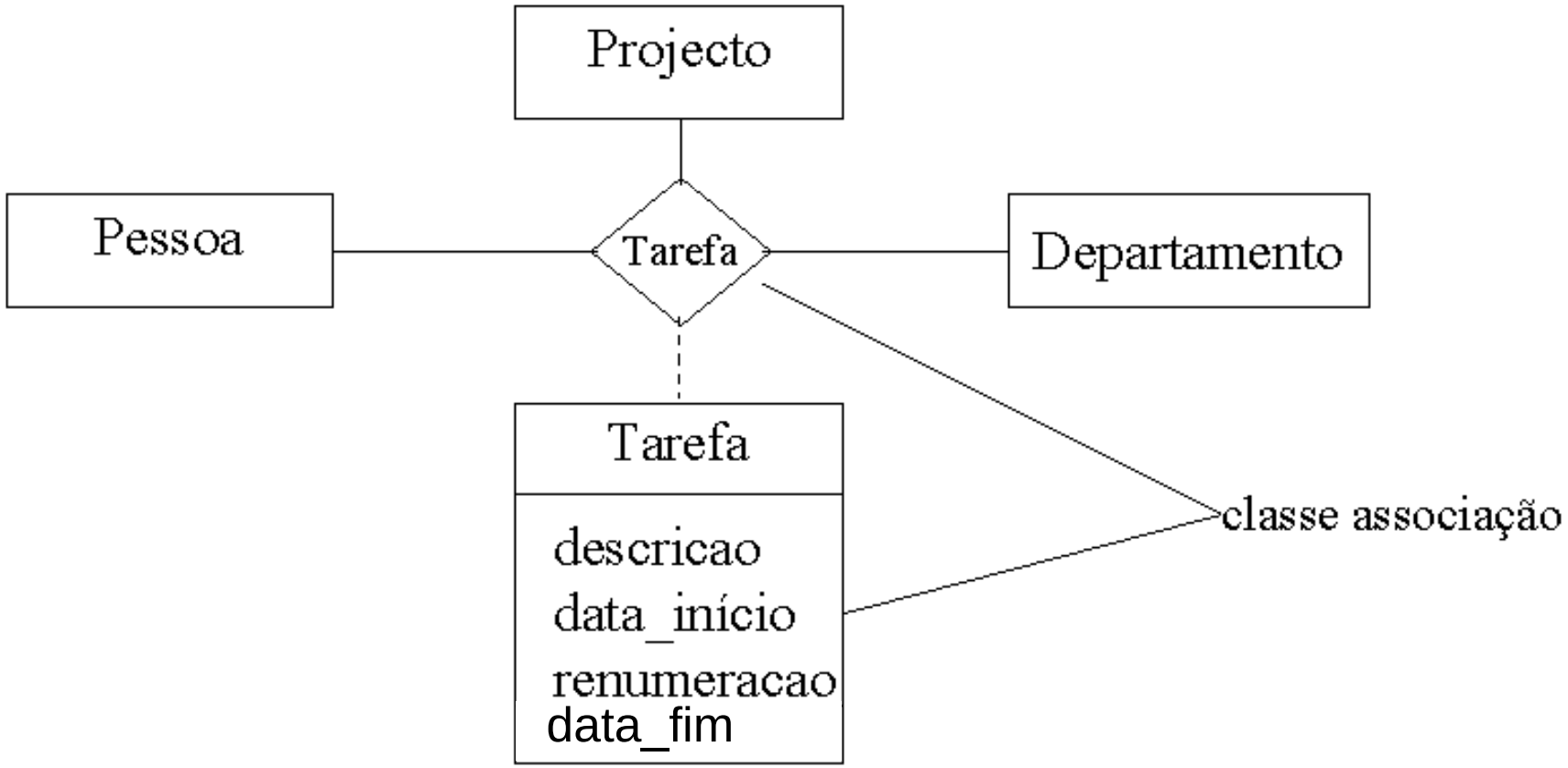
Associações Unárias ou Reflexivas

- Uma associação diz-se reflexiva quando estabelece uma relação duma classe consigo própria
- Este tipo de associação acontece quando as ocorrências de uma classe desempenham diferentes papéis



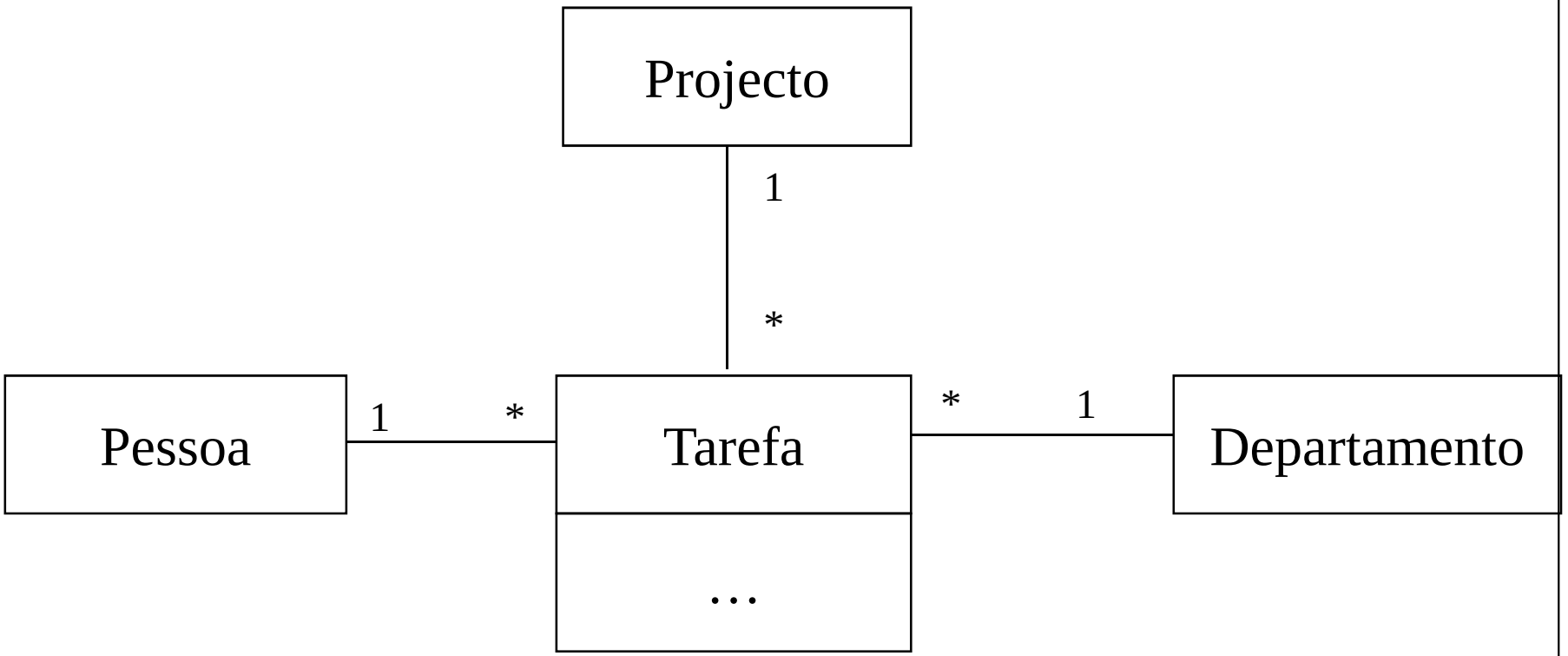
Associações N-Árias (N>= 3)

As associações N-Árias (N>= 3) são relativamente pouco comuns



Associações N-Árias (N>= 3)

As associações N-árias podem geralmente ser transformadas em várias associações binárias entre a classe-associação e as restantes classes participantes



Como identificar as classes

- **Identificar as classes:**
  - » a descrição de um problema refere normalmente objectos concretos, mas são as abstracções do mundo real (os objectos é que são do mundo real, não as abstracções...) que permitem detectar aspectos comuns a vários objectos semelhantes
  - » normalmente, inicia-se o processo sublinhando os substantivos (na maior parte dos casos correspondem a abstracções que vão originar classes)
  - » escolher muito bem os nomes (num diagrama de classes, o nome é muitas vezes a informação visível sobre uma classe)

Falsas Classes

- Um dos erros mais perigosos e mais frequentes em OO é identificar como "classe" algo que realmente não o é (atenção a substantivos que não correspondem a classes)

- Um desses erros é descrever uma classe pelo que ela faz: "esta classe desenha figuras". Uma **classe não faz**. Uma classe **é** uma definição (descreve um conjunto de objectos através dos seus atributos e operações)

- Um conceito, para ser considerado "classe", tem que possuir riqueza semântica (vários atributos e várias operações)

Em síntese...

» As características mais importantes das classes são:
  – **devem ser relevantes**
  – **ter diversas ocorrências**
  – **ser dotadas de riqueza semântica (vários atributos e várias operações)**

» As características mais importantes dos atributos são:
  – **devem ser relevantes**
  – **elementares**
  – **atómicos em cada ocorrência da classe e**
  – **variáveis de ocorrência para ocorrência da classe**

Método de Booch

Booch propôs um método simples para identificar classes, que consiste em:

- sublinhar os substantivos na descrição de um problema
- distinguir os que são classes dos que são atributos

Exemplo:

"Quando o comboio se aproxima da estação, a sua velocidade diminui"

Classes prováveis: "comboio" e "estação";
Atributos prováveis: "velocidade" é por certo um dos atributos de "comboio"...