III - Noções de Análise e Desenvolvimento Orientado por Objectos

António Palma dos Reis
Aristides Sousa Mendes
Filipa Pires da Silva
Winnie Picoto

Índice

- Introdução
- Conceitos Básicos de Orientação por Objectos (OO)
- Classes e Relacionamentos entre Classes
- Propriedades Fundamentais das Classes

Introdução

A área de **OO** emergiu da convergência de muitos dos resultados de investigação noutras áreas:

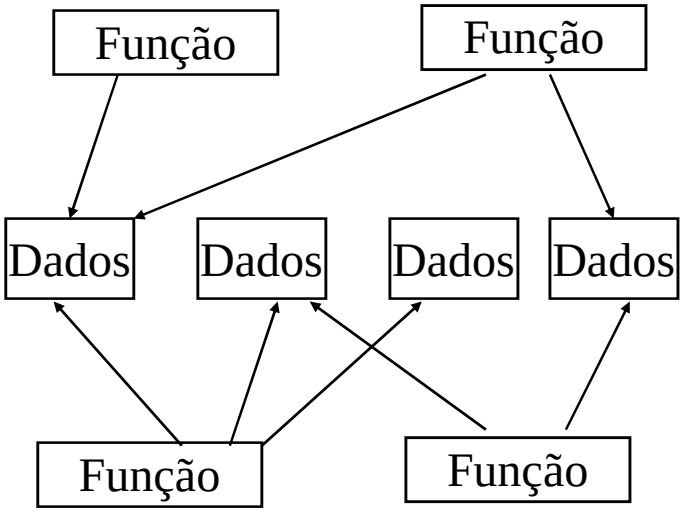
- Inteligência artificial;
- Engenharia de Software;
- Teorias de abstracção de dados;
- Gestão de Dados Complexos (listas, arrays, OLEs, comportamento).

O que é a Orientação por Objectos? (1/2)

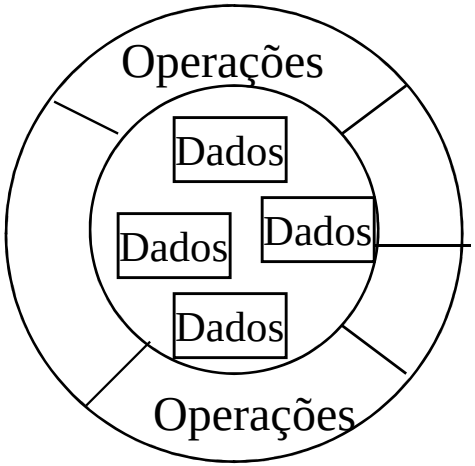
- O paradigma OO baseia-se numa nova forma de analisar o mundo que reproduz a forma como o ser humano se apercebe e expressa a realidade que o rodeia
  - O mundo é classificado e subdivido em diferentes objectos, com base nas diferenças e semelhanças existentes ao nível das características e comportamentos dos mesmos
- Os objectos existem no mundo real;
- A abordagem OO permite suportar:
  - tipos de dados complexos,
  - modelos de dados semanticamente mais ricos
  - reutilização

O que é a Orientação por Objectos? (2/2)

Programação Tradicional
(funções e dados não estão unificados)



Programação OO
(operações e dados estão unificados)



Abstracção de Objectos (1/3)

- A **abstracção -** é a forma de descrever algo pelas suas propriedades essenciais, ignorando o detalhe, que é desnecessário;
- Permite representar os objectos;
- Pode não ser única (o conjunto de propriedades "essenciais" pode variar consoante o fim a que se destina);
- A escolha da abstracção mais adequada é talvez a parte mais difícil da programação orientada por objectos.

A Abstracção de objectos conduz ao conceito de Classe

Abstracção de Objectos (2/3)

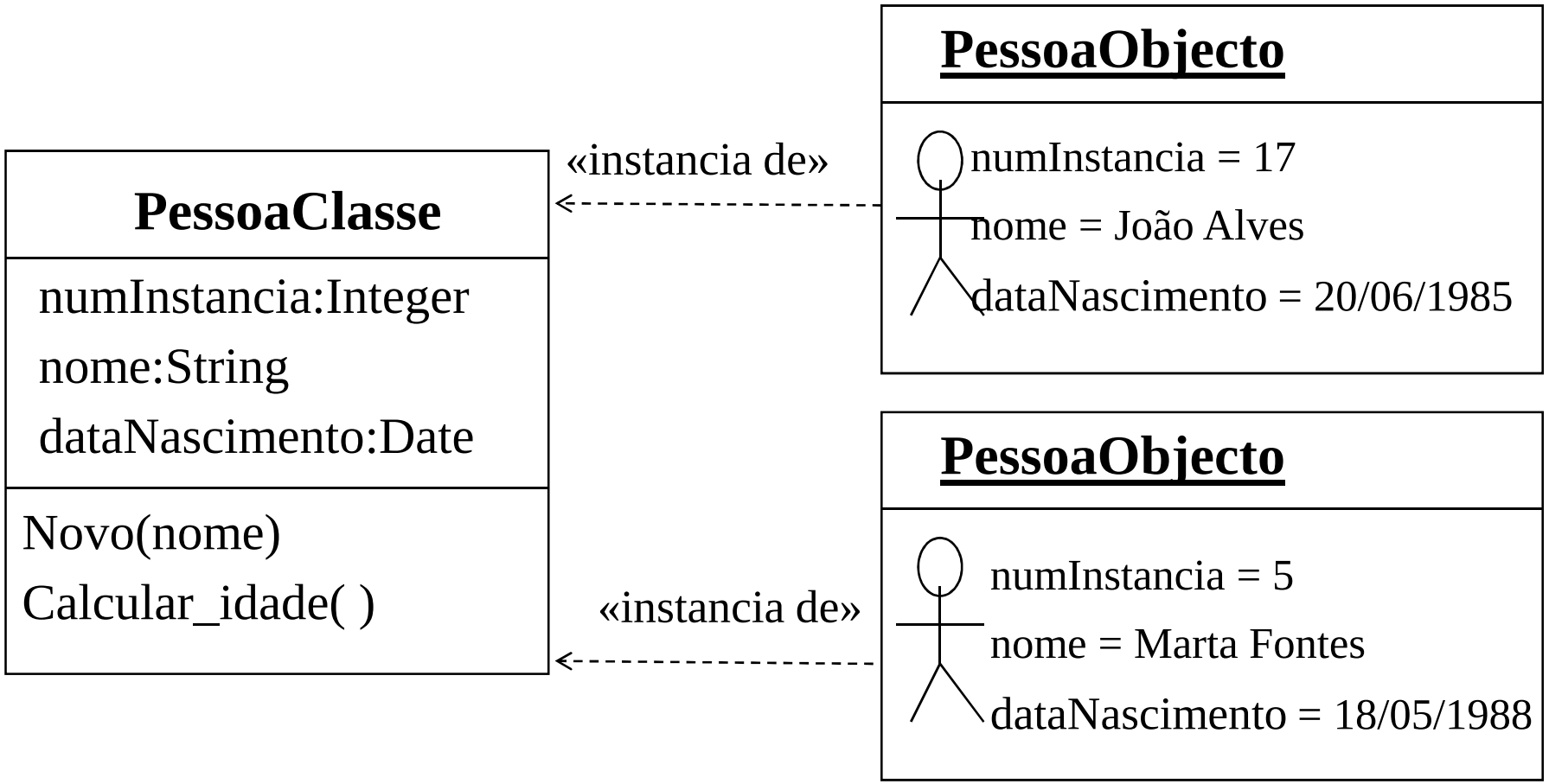
• Uma classe agrega todos os objectos que partilhem as mesmas características, comportamento, relações e semântica.

•Pode dizer-se que:

- •uma classe é uma abstracção de objectos semelhantes
- •um objecto é uma instância de uma classe

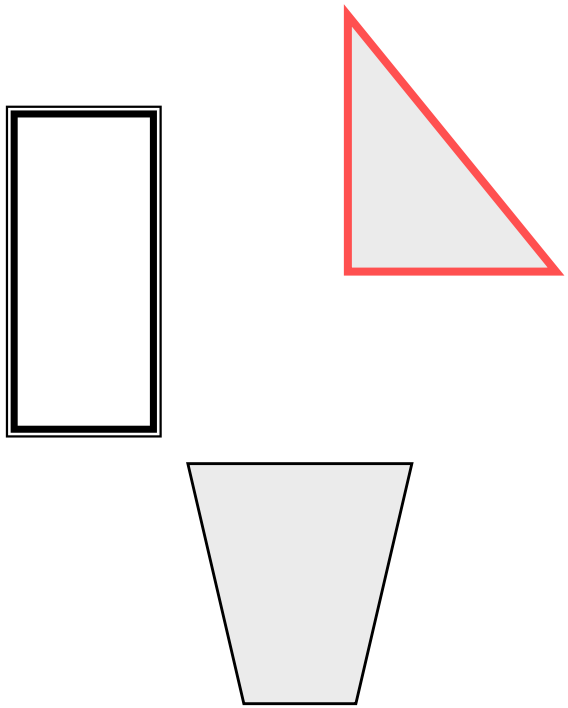
• As classes têm Atributos (resultado da abstracção dos dados) e Operações ou Serviços (resultado da abstracção dos métodos).

Abstracção de Objectos (3/3)



Exemplo de Classe

Objectos “Polígonos”



Abstracção para

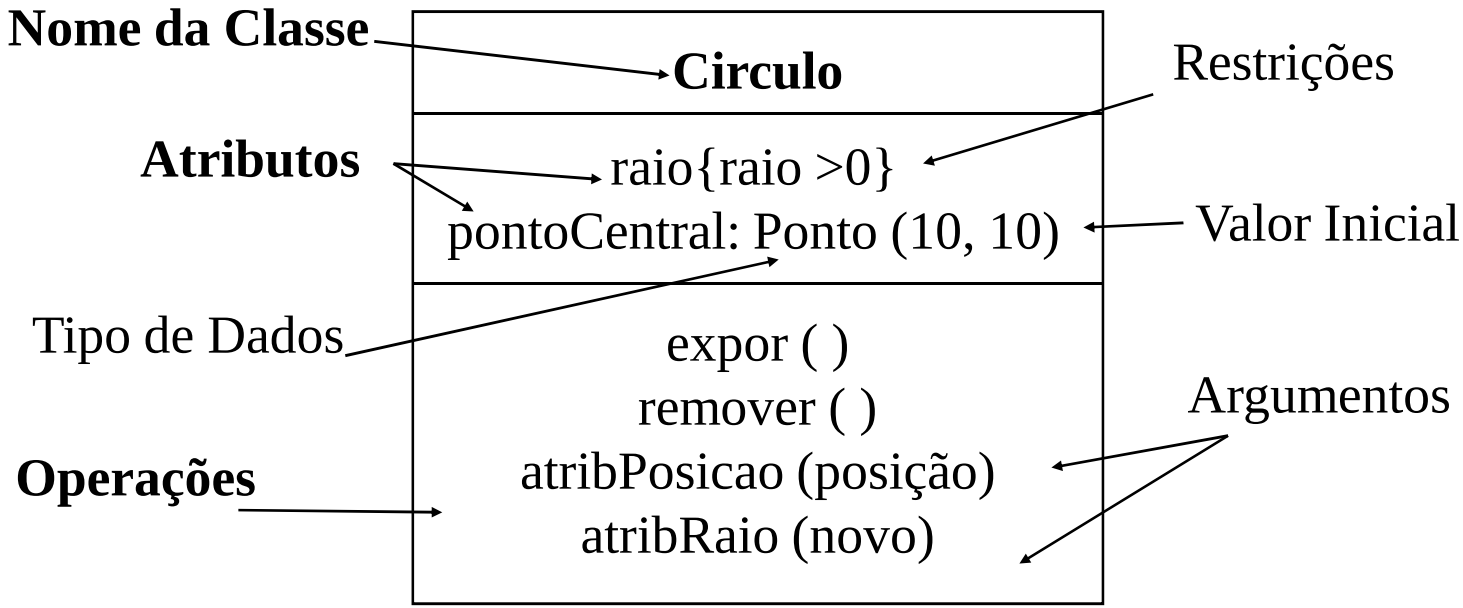
Classe “Polígono”
Atributos: vértices cor do rebordo cor interior ...
Operações: zoom( ) mover( ) apagar() ...

Representação Formal duma Classe (1/3)

Para representarmos um círculo no ecrã precisamos das seguintes características:

***atributos*:** o seu raio e respectiva posição no ecrã (x,y);
operações: a possibilidade de ser exposto, removido, redimensionado;
***restrições*:** o valor do raio tem que ser positivo (raio >0);
relacionamentos: neste exemplo o círculo não possui relacionamentos.

Uma classe é representada por um rectângulo com uma, duas ou três secções



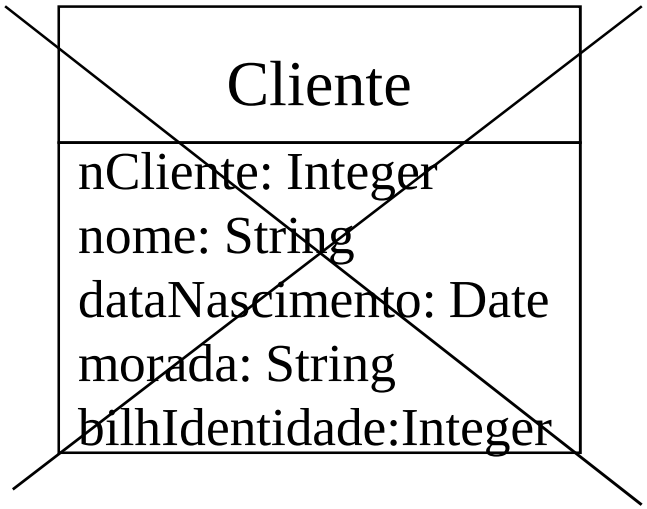
Representação Formal duma Classe (2/3)

- **Nome da Classe**: distingue a identidade da classe
  - » São substantivos retirados do vocabulário do domínio
  - » Devem ser únicos
  - » Convenção: Primeira letra de todas as palavras capitalizada

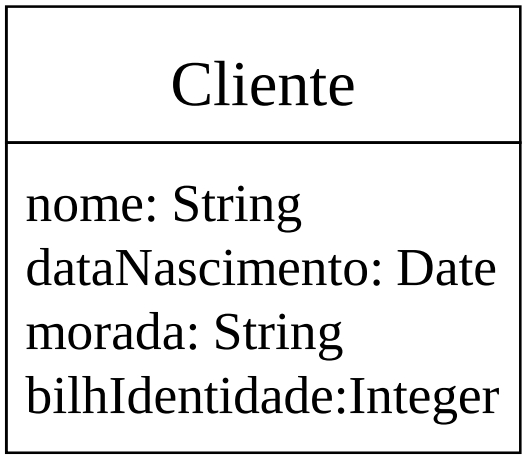
- **Atributo**: característica que os objectos possuem e que é representada por um valor de dados.
  - » O nome do atributo é obrigatório e tem de ser único no contexto da classe onde é definido.
  - » Convenção: Primeira letra de todas as palavras capitalizada, com excepção da primeira
  - » O tipo de atributo determina o tipo de informação que pode ser guardada no atributo, por exemplo “nome: String”, “dataNascimento: Date”

Representação Formal duma Classe (3/3)

- 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)



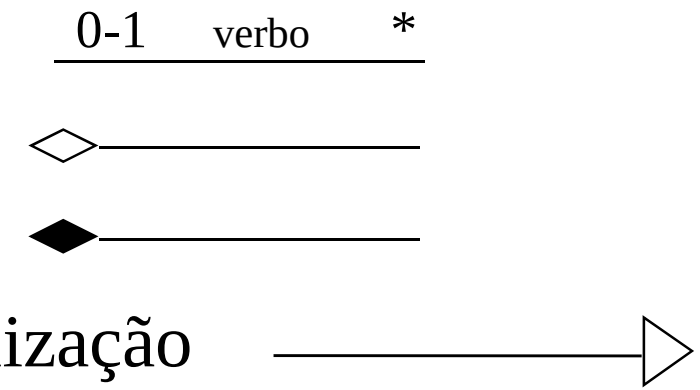
Errado



Certo

Relacionamentos entre Classes

- Tipos de relacionamentos que se podem estabelecer entre classes:
  - » Associação
  - » Agregação
  - » Composição
  - » Generalização/Especialização



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

Adornos dos Relacionamentos (1/2)

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

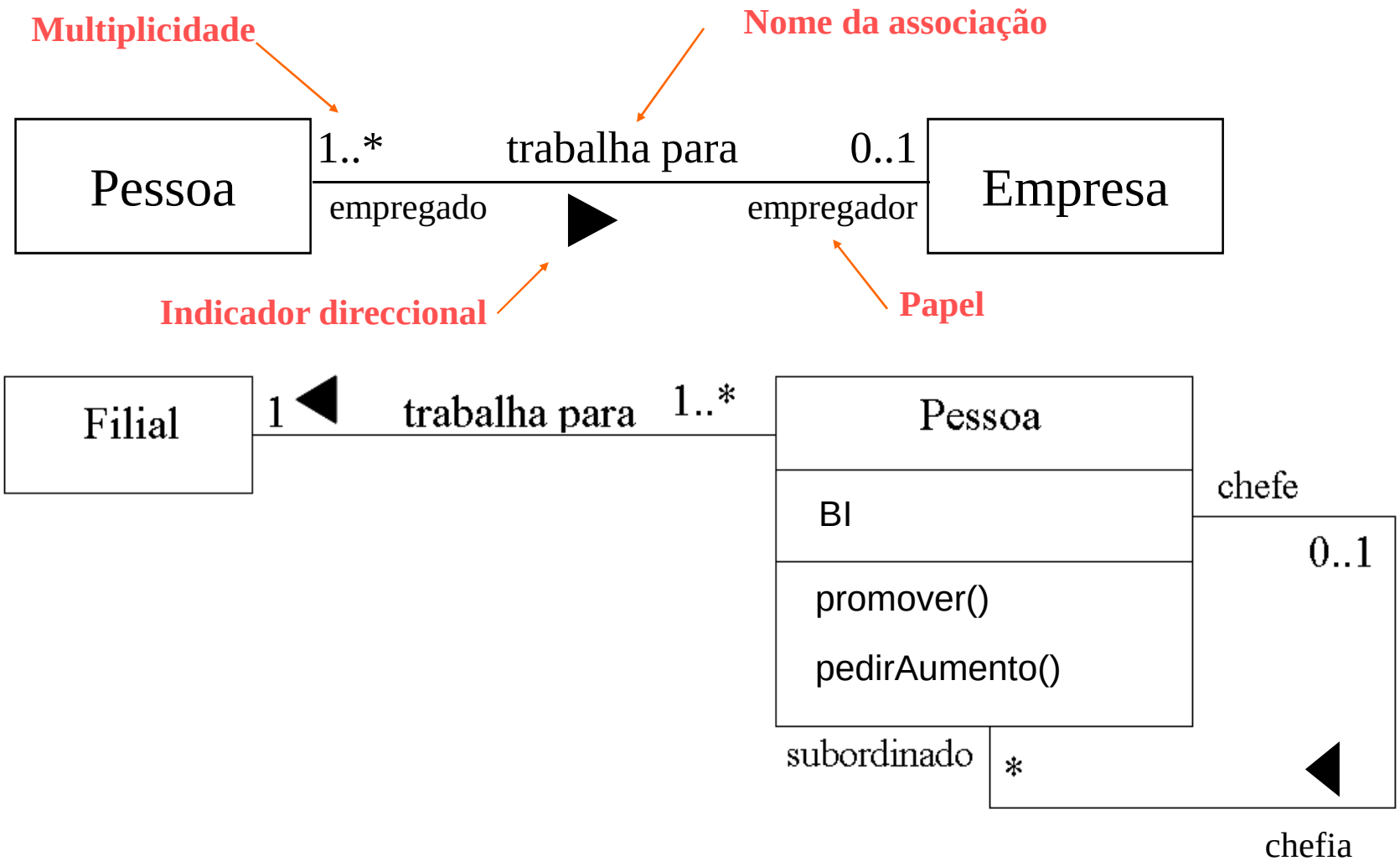
Adornos dos Relacionamentos (2/2)

- Multiplicidade: 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)
  - » Note-se que o limite mínimo da multiplicidade descreve a obrigatoriedade do relacionamento

- 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

Exemplos de Relacionamentos entre Classes (1/2)

» Associação

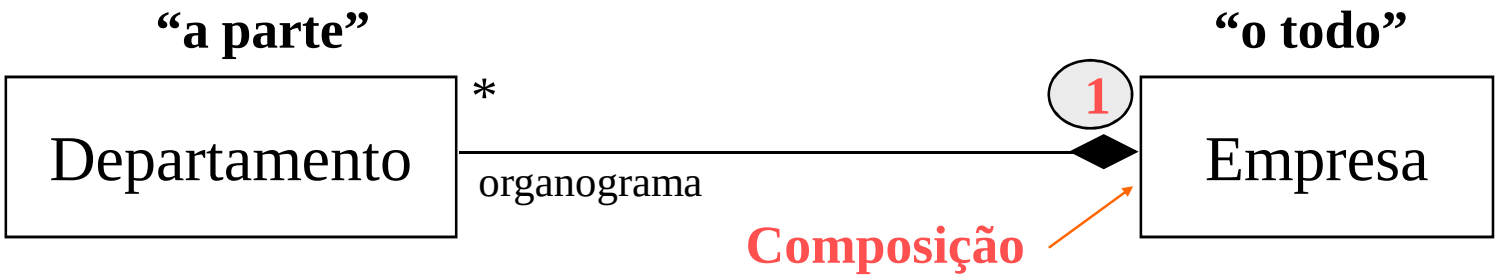


Exemplos de Relacionamentos entre Classes (2/4)

» Agregação

Importante: A parte pode permanecer sem o todo!

» Composição

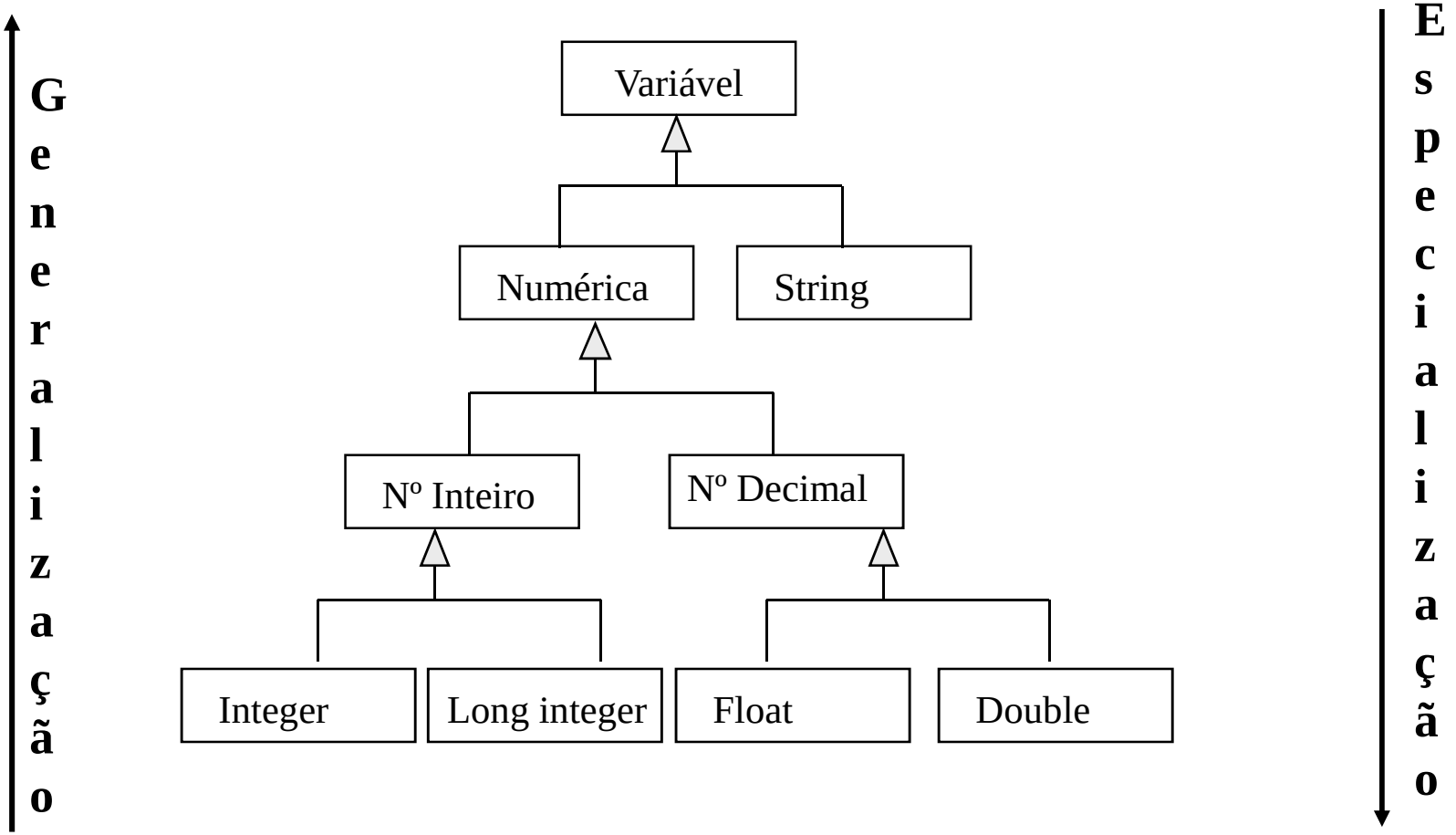


Importante: As partes não podem existir sem o todo!

Exemplos de Relacionamentos entre Classes (3/4)

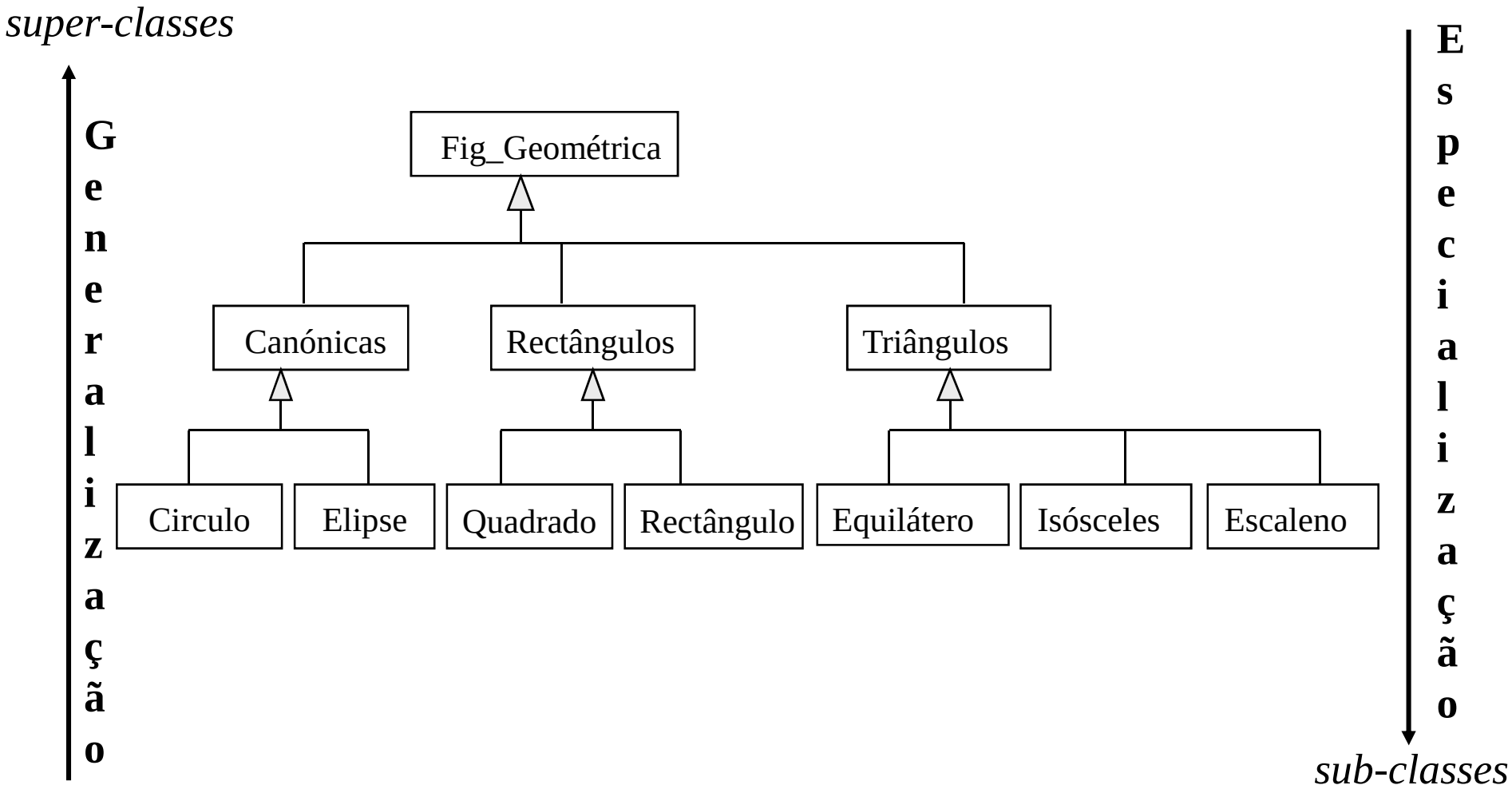
» Herança

super-classes



Exemplos de Relacionamentos entre Classes (4/4)

» Herança



Troca de Mensagens

- Os objectos são unidades independentes que cooperam e interactuam através de mensagens que enviam uns aos outros;

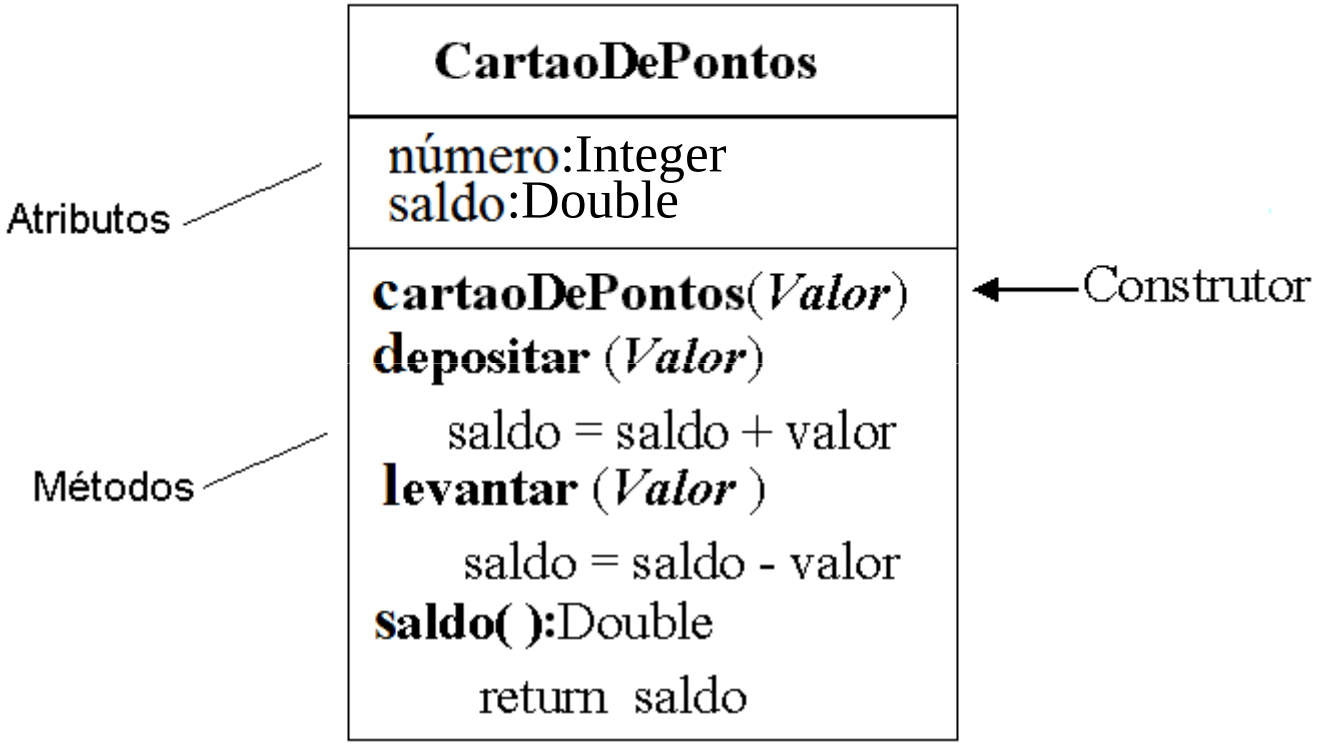
- As mensagens trocadas entre objectos representam a invocação de um serviço (operação) disponibilizado por um objecto, com o objectivo de despoletar uma acção ou actividade

Métodos

- A interface de um objecto (o seu protocolo) é definido em termos de métodos (acções desencadeadas pelas mensagens) a que ele sabe responder;
- Um método é desencadeado pela recepção duma mensagem;
- A maioria dos métodos responde a perguntas ou alteram o estado do objecto. Na prática implementam um serviço e/ou funcionalidade que pode ser requerido por qualquer objecto que faça parte do universo do problema
- Cada método pode incluir várias instruções de envio de mensagens;

Exemplo de Mensagens (1/2)

- Definição da classe *CartãoDePontos*

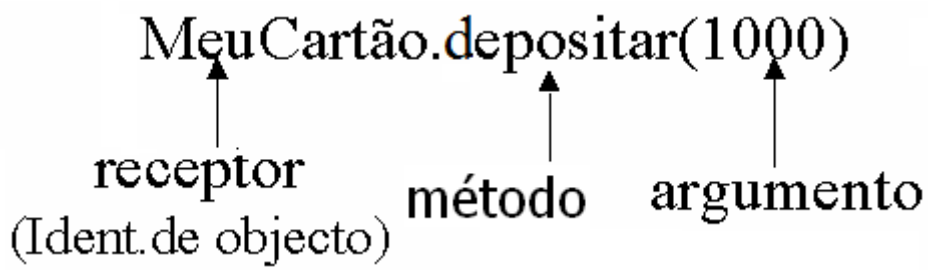


- Criação de um cartão (instância da classe CartãoDePontos)

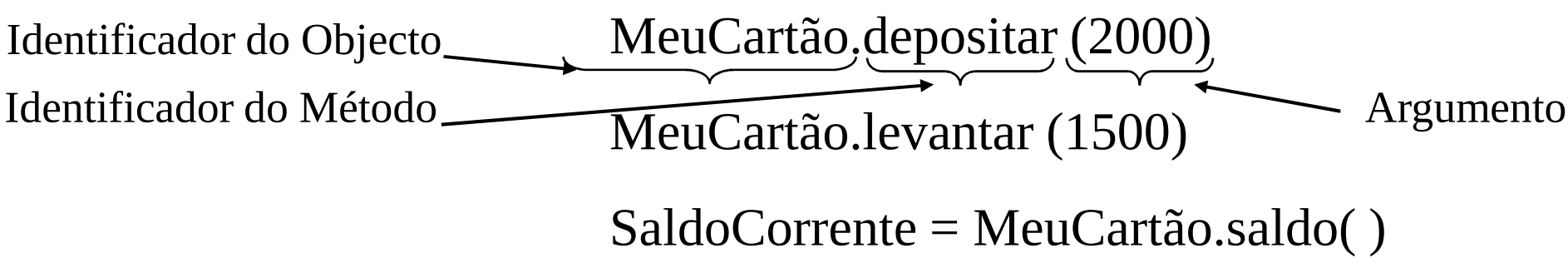
MeuCartão <- CartãoDePontos.new

Identificador de classe Construtor

Exemplo de Mensagens (2/2)



•Uso do MeuCartão



Propriedades da OO

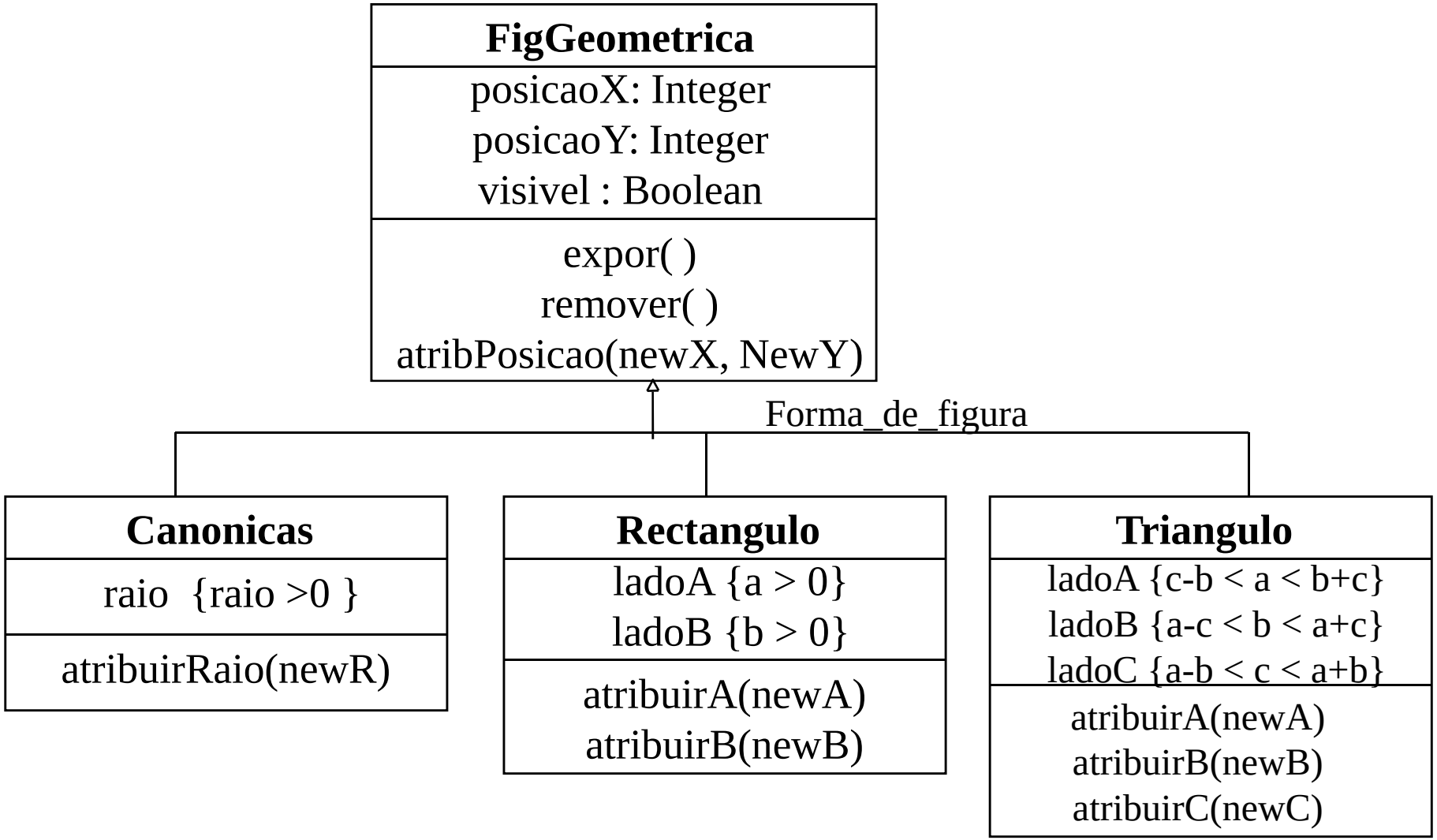
Encapsulamento – as classes combinam os atributos e as operações numa mesma unidade. Os Atributos e a implementação das operações ficam escondidos dos utilizadores, não sendo necessário conhecê-los para se aceder aos objectos;

Herança – as classes podem ser organizadas hierarquicamente e assumir (“herdar”) as propriedades das super-classes;

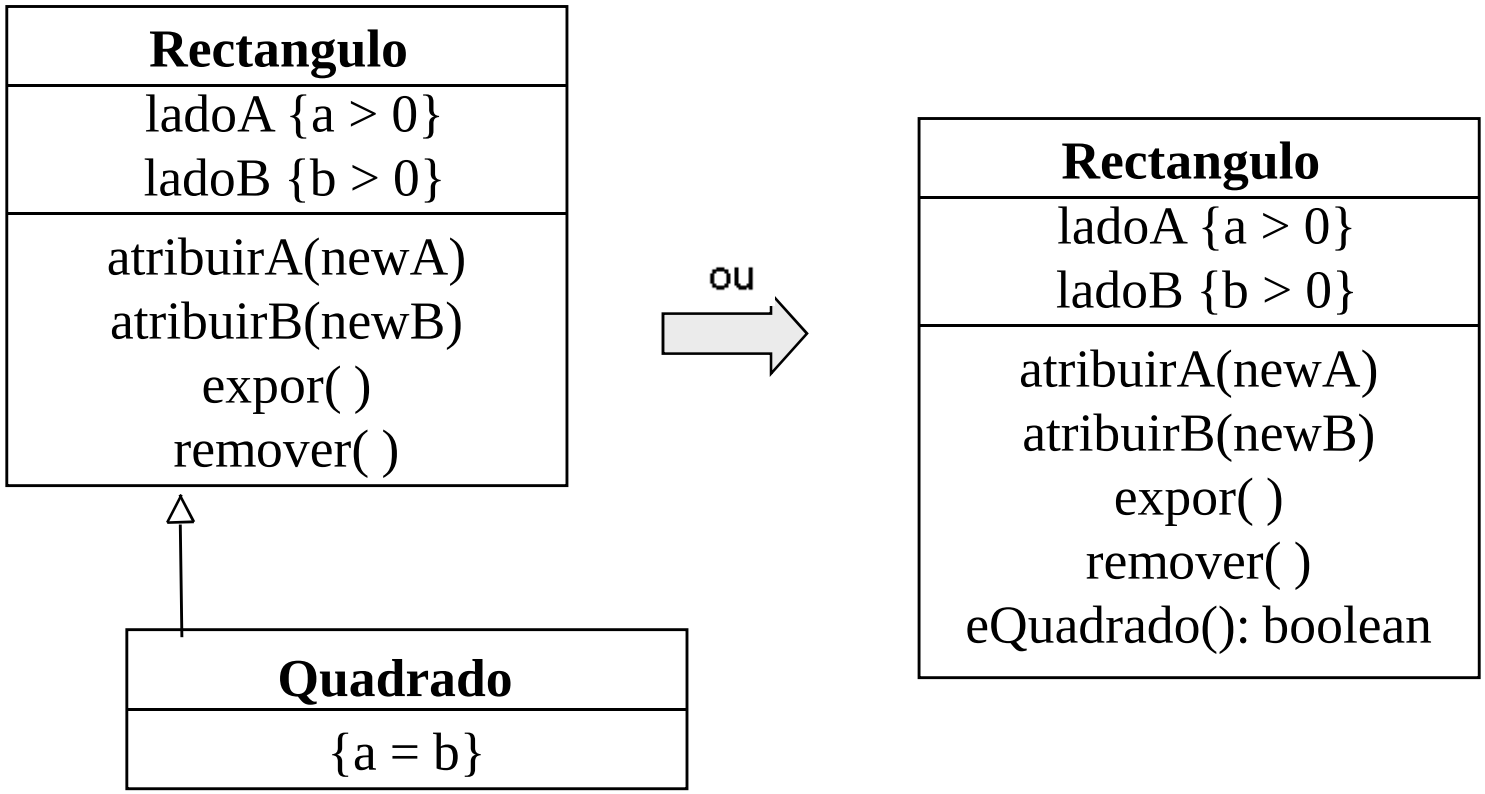
Polimorfismo – propriedade que permite que objectos de classes diferentes reajam diferentemente a mensagens com o mesmo método;

Abstracção – representação concisa de ideias ou da essência de objectos incidindo sobre as suas características essenciais

Estruturar Hierarquia de Características



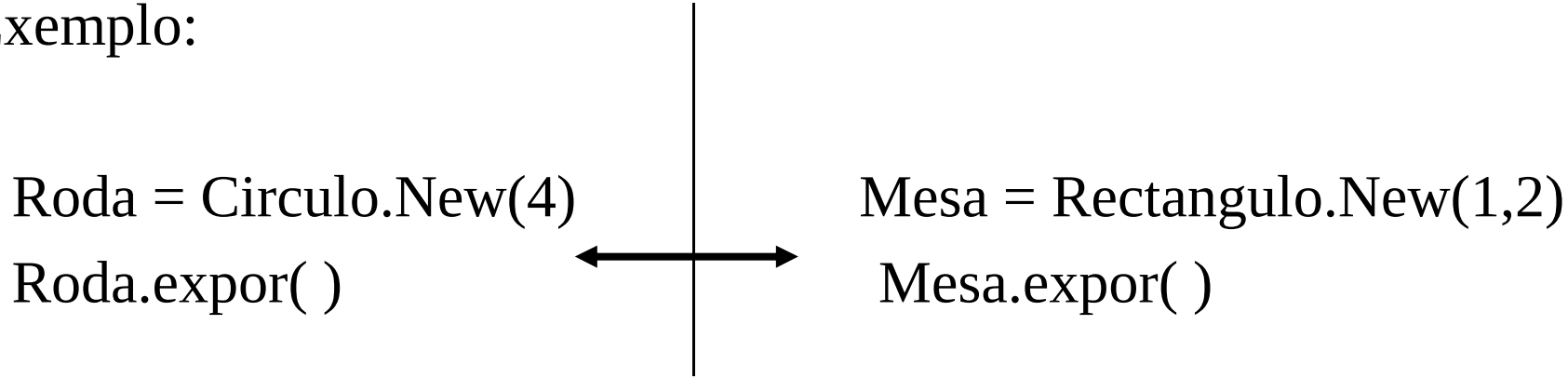
Herança: Restrições e Alternativas



Polimorfismo

- **Polimorfismo -** propriedade que permite que objectos de classes diferentes reajam diferentemente a mensagens com o mesmo método.

Exemplo:



Construção do Modelo de Análise

- 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);

- Se uma "classe" tem apenas uma função na interface, então é apenas uma função com rótulo de classe;

- Um ente, 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 dotados 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.

O método consiste em:

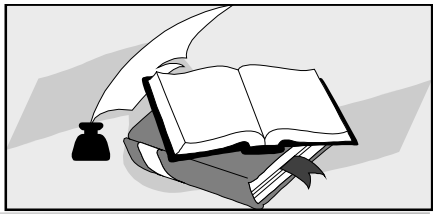
- sublinhar os substantivos na descrição de um problema,
- distinguir os que são classes dos que são atributos.

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"...

Bibliografia Utilizada :



Pereira, José Luís Mota (2001) *Tecnologia de Bases de Dados*, Editora de Informática LDA.

Oestereich, Bernd, ***Developing Software With UML***, Addison Wesley, second edition, 2002.

Rumbaugh, J., Blaha, M., Premerlani. W., Eddy, F., Lorensen, W.; ***Object-Orientes Modeling and Design***, Prentice-Hall Inc., USA., 1991.

Sampaio, F.(1996-2000). *Transparências, Notas e Exercícios para a disciplina Sistemas de Gestão de Bases de Dados*. Pós-graduação em Sistemas e Tecnologias da Informação para as Organizações, ISEG, Lisboa.

Sampaio, F. (2004-2008) *Transparências, Notas e Exercícios para a Unidade Curricular de Sistemas de Informação*, ISEG/FMH/UTL, ISEG/FMH, Lisboa.

Silva, Alberto, Videira, Carlos ***UML Metodologias e Ferramentas CASE***, Centro Atlântico, Publishing Ltd.