

# *Orientação a Objetos: Conceitos Básicos*

---

## *Sumário*

---

- ☞ Introdução
- ☞ Histórico
- ☞ Vantagens
- ☞ Conceitos Básicos
  - Decomposição
  - Abstração
  - Encapsulamento
  - Classificação
  - Herança
  - Polimorfismo

## *Histórico de OO*

---

- ☛ A OO surgiu no final da década de **60**, quando cientistas dinamarqueses criaram a linguagem Simula (*Simulation Language*)
  - **1967** - Linguagem de Programação Simula-67-  
*conceitos de classe e herança*
- ☛ O termo Programação Orientada a Objeto (POO) é introduzido com a linguagem Smalltalk (**1983**)
- ☛ Fim dos anos **80**: **Paradigma OO**

## *Histórico de OO*

---

- ☛ Surgiram linguagens híbridas:
  - C++ (**1986**), Object-Pascal (**1986**)
- ☛ Surgiram diversos Métodos de Análise e Projeto OO
  - CRC (*Class Responsibility Collaborator*, Beecke e Cunningham, **1989**)
  - OOA (*Object Oriented Analysis*, Coad e Yourdon, **1990**)
  - Booch (**1991**)
  - OMT (*Object Modeling Technique*, Rumbaugh, **1991**)
  - Objectory (Jacobson, **1992**)
  - Fusion (Coleman, **1994**)
  - UML (*Unified Modeling Language*, **1997**)

## *Vantagens de OO* (Maldonado, 2002)

---

- ☛ **Abstração de dados:** os detalhes referentes às representações das classes serão visíveis apenas a seus atributos
- ☛ **Compatibilidade:** as heurísticas para a construção das classes e suas interfaces levam a componentes de software que são fáceis de se combinar

## *Vantagens de OO* (Maldonado, 2002)

---

- ☛ **Flexibilidade:** as classes delimitam-se em unidades naturais para a alocação de tarefas de desenvolvimento de software
- ☛ **Reutilização:** o encapsulamento dos métodos e representação dos dados para a construção de classes facilitam o desenvolvimento de software reutilizável, auxiliando na produtividade de sistemas

## Vantagens de OO (Maldonado, 2002)

---

- ☞ **Extensibilidade**: facilidade de estender o software devido a duas razões:
  - herança: novas classes são construídas a partir das que já existem;
  - as classes formam uma estrutura fracamente acoplada o que facilita alterações;
- ☞ **Manutenibilidade**: a modularização natural em classes facilita a realização de alterações no software.

## Vantagens (Maldonado, 2002)

---

- ☞ Melhora de **comunicação** entre desenvolvedores e clientes
- ☞ Redução da **quantidade de erros** no sistema, diminuindo o tempo nas etapas de codificação e teste
- ☞ Maior dedicação à fase de análise, preocupando-se com a **essência do sistema**
- ☞ **Mesma notação** é utilizada desde a fase de análise até a implementação

## *Conceitos Básicos*

---

- ☛ Orientação a Objetos (OO) é uma abordagem de que procura explorar o nosso lado intuitivo.
- ☛ Os objetos da computação são análogos aos objetos existente no mundo real.
  - No enfoque de OO, os átomos do processo de computação são os objetos que trocam mensagens entre si.
  - Essas mensagens resultam na ativação de métodos, os quais realizam as ações necessárias.
  - Os objetos que compartilham uma mesma interface, ou seja, respondem as mesmas mensagens, são agrupados em classes.

(Maldonado, 2002)

## *Paradigma Orientado a Objetos*

---

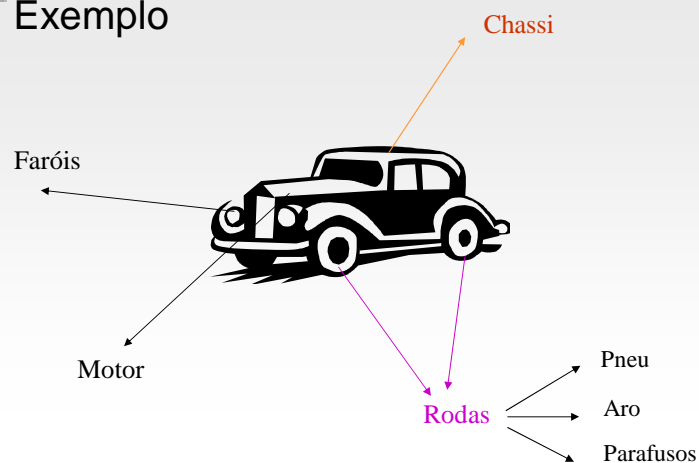
- ☛ Estratégias para se administrar a complexidade
  - Decomposição (Modularidade)
  - Abstração
  - Encapsulamento
  - Classificação (Hierarquias)

## Decomposição

- “Dividir para conquistar”
- Decomposição sucessiva de problemas em subproblemas de mais fácil tratamento
- A decomposição pode ser feita sob o ponto de vista
  - de um elemento e suas partes
  - de um procedimento e suas etapas

## Decomposição

### Exemplo



## Decomposição

### Exemplo

#### Fazer um bolo:

1) Fazer a massa

Misturar:

2 ovos

4 colheres de açúcar

1 copo de leite

Bater bem..

2) Fazer o recheio

Colocar no fogo:

água

Açúcar

Ovos

Mexer até ferver

3) Fazer a cobertura

Colocar no fogo:

Leite

Chocolate

Mexer até engrossar

## Decomposição

### Estruturas de Dados

#### Registro Professor

```
{  
    String nome;  
    String matrícula;  
}
```

## *Decomposição*

---




### Funções e Procedimentos

Procedimento Desenhar Quadrado (Coordx, Coordy, lado, cor)

```
{  
    DesenharLados(Coordx, Coordy, lado, cor)  
    PintarFundo(Coordx, Coordy, lado, cor)  
}
```

## *Abstração*

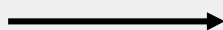
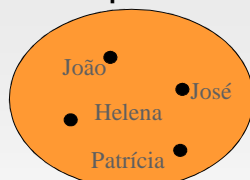
---

-  Reconhecimento de similaridades entre certos objetos, situações ou processos no mundo real e a decisão de concentrar nessas similaridades e ignorar, durante um certo tempo, as diferenças
-  Uma descrição simplificada de um sistema que enfatiza alguns detalhes ou propriedades, suprimindo outras
-  Uma abstração denota as características essenciais de um objeto que o distingue de todos os demais, relativamente à perspectiva de quem o vê

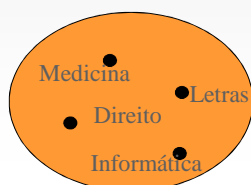


## Abstração

### Exemplo



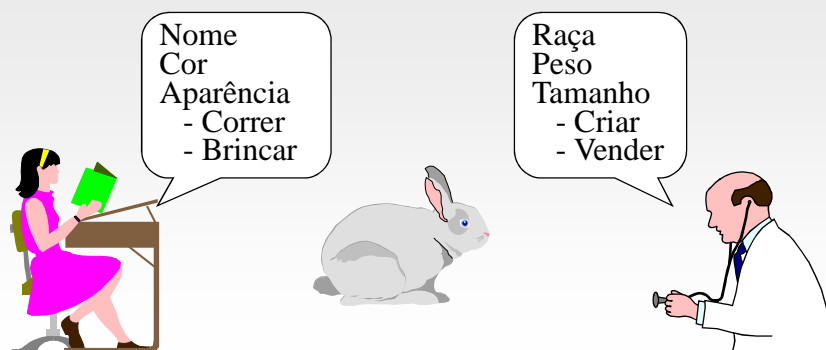
**Aluno**  
Nome  
Matrícula  
Telefone  
Data de Nascimento  
Endereço



**Curso**  
Nome  
Código  
Número de Créditos

## Abstração

### A abstração depende do ponto de vista:



## Classificação

Princípio através do qual é possível agrupar objetos e relacionar estes grupos segundo suas características comuns



Exemplo

- Animal
  - Anfíbio
    - ...
  - Réptil
  - Mamífero
    - Roedor
      - ...
  - Ave
- Vegetal

## Objeto

Definição:

- Um conceito, uma abstração com significado específico em um contexto

Propósito:

- representar uma entidade do mundo real

Objetos possuem:

- *Identidade*
- Conjunto de *características* que determinam seu *estado*
- *Comportamento* específico definido por um conjunto de ações

## Exemplos



*Identidade:* 'Beija-flor Biju'

*Características:*

penas azuis  
bico fino  
vôo rápido

*Comportamento:*

voar  
piar



*Identidade:* 'Pessoa Mário'

*Características:*

olhos pretos  
nasceu em 16/02/70  
pesa 70kg  
mede 1,70m

*Comportamento:*

andar  
falar  
comer  
rir

## Exemplos



*Identidade:* 'Telefone da minha casa'

*Características:*

azul  
número 576-0989  
tone

*Comportamento:*

tocar  
discar

*Identidade:* 'ônibus da escola'

*Características:*

cor amarela  
placa LXY 7684  
30 assentos  
a diesel

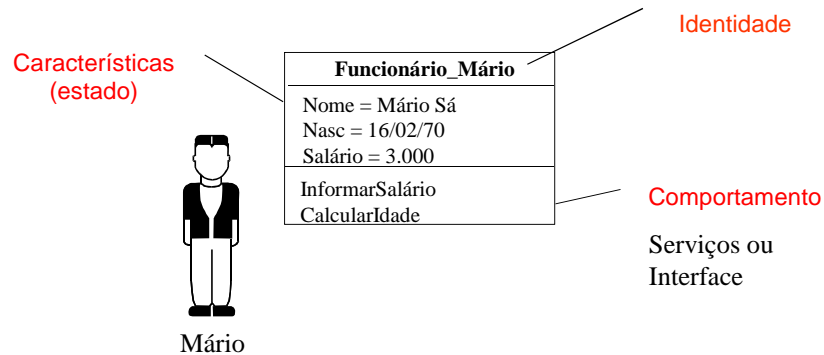
*Comportamento:*

frear  
andar  
correr  
buzinar  
acelerar



## Objeto

### Representação



## Classe

### Definição:

- Abstrações utilizadas para representar um conjunto de objetos com *características e comportamento idênticos*

### Uma classe pode ser vista como uma “fábrica de objetos”

### Objetos de uma classe são denominados “*instâncias*”

- Todos os objetos são instâncias de alguma classe
- Todos os objetos de uma classe são idênticos no que diz respeito a sua interface e implementação

## Exemplo

Pássaro



### Características:

cor das penas  
formato do bico  
velocidade de vôo

Identidade: 'Beija-flor Bijú'

### Características:

cor das penas: azuis  
formato do bico: fino  
velocidade de vôo: rápida

### Comportamento:

voar  
piar

### Comportamento:

voar  
piar



Identidade: 'Minha pomba'

### Características:

cor das penas: cinza  
formato do bico: curto  
velocidade de vôo: média

### Comportamento:

voar  
piar

## Exemplo

Telefone



### Características:

cor  
número  
discagem

Identidade: 'Telefone da minha casa'

### Características:

cor: azul  
número: 576-0989  
discagem: tom

### Comportamento:

tocar  
discar

### Comportamento:

tocar  
discar



Identidade: 'Meu celular'

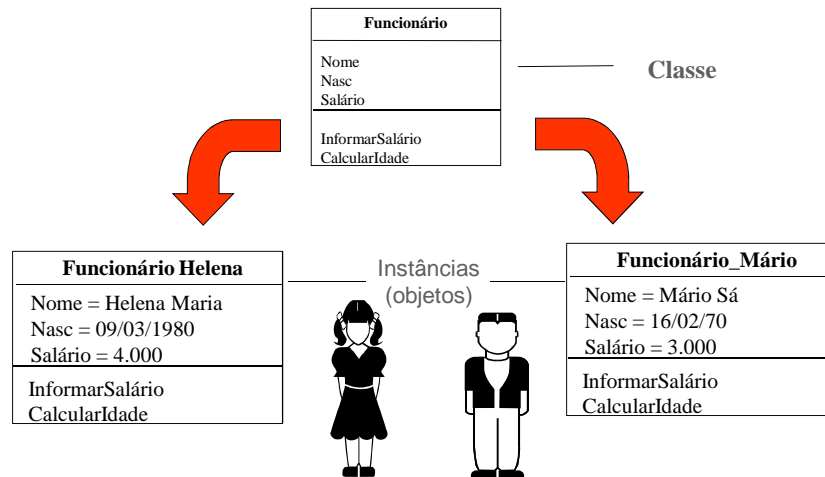
### Características:

cor: preta  
número: 99193467  
discagem: tom

### Comportamento:

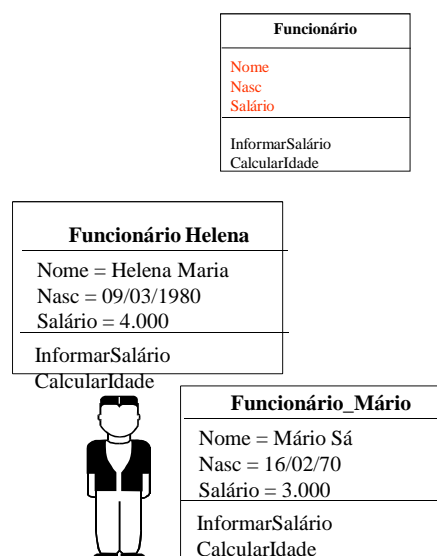
tocar  
discar

## Classe



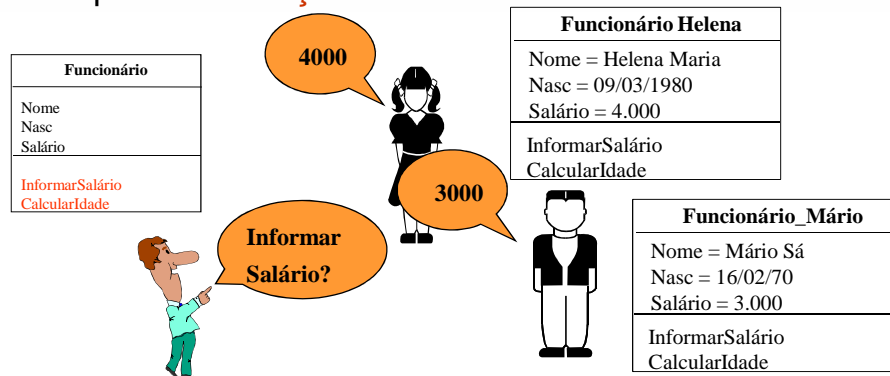
## Atributos

- Descrevem as **características** das instâncias de uma classe
- Seus valores definem o **estado** do objeto
- O estado de um objeto pode mudar ao longo de sua existência
- A identidade de um objeto, contudo, nunca muda



## Serviços/Métodos

- Representam o **comportamento** das instâncias de uma classe
- Correspondem às **ações** das instâncias de uma classe



## Serviços/Métodos

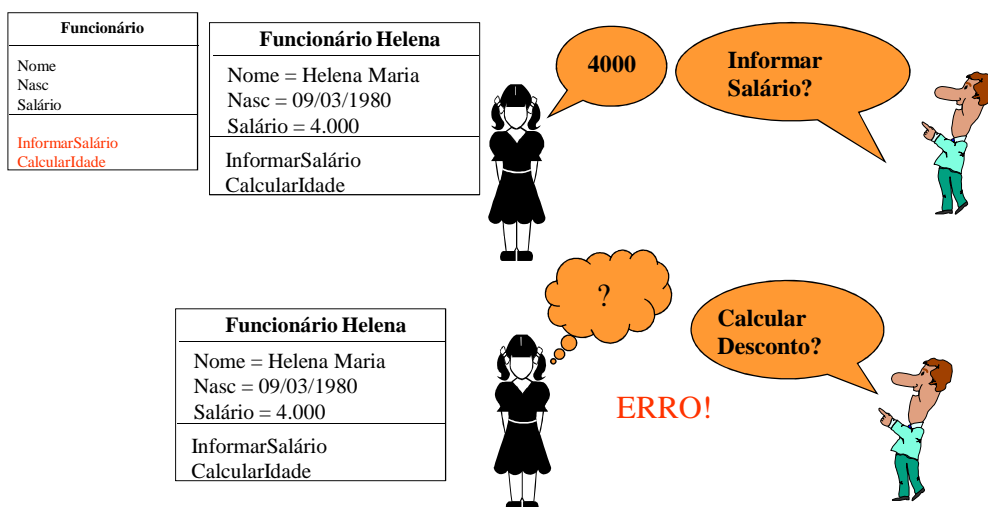
- Um método é a **implementação** de uma operação
  - possuem argumentos, variáveis locais , valor de retorno, etc
- Alguns métodos especiais:
  - Construtores – criam objetos de uma classe
  - Destrutores – destroem objetos de uma classe

## Mensagens

- Objetos são entidades independentes que necessitam se comunicar
  - Para obter informações ou ativar o comportamento de objetos, é preciso enviar-lhes **mensagens**
  - Ao receber uma mensagem, o objeto busca em seu protocolo um **método** que irá **responder** a tal mensagem
- Objetos só reagem a mensagens que fazem parte das ações do protocolo de sua classe

✉ Troca de mensagens: Paradigma de comunicação entre objetos

## Mensagens





## Encapsulamento

### Objetivos

- Restringir o escopo de informação
- Atingir legibilidade, manutenibilidade e reuso



### Princípio muito explorado na orientação a objetos

- Concentra a visão na parte externa do objeto, omitindo suas características internas
- Os dados de uma classe só podem ser manipulados por métodos da classe

## Objeto

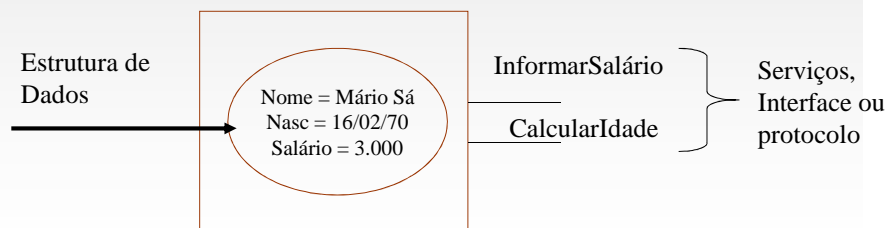
### Implementação

- Interface Visível
  - ➔ Conjunto de operações públicas
- Parte encapsulada (escondida)
  - ➔ Implementação
    - Estrutura de dados – Estado
    - Implementação de suas operações - Métodos

## Objeto

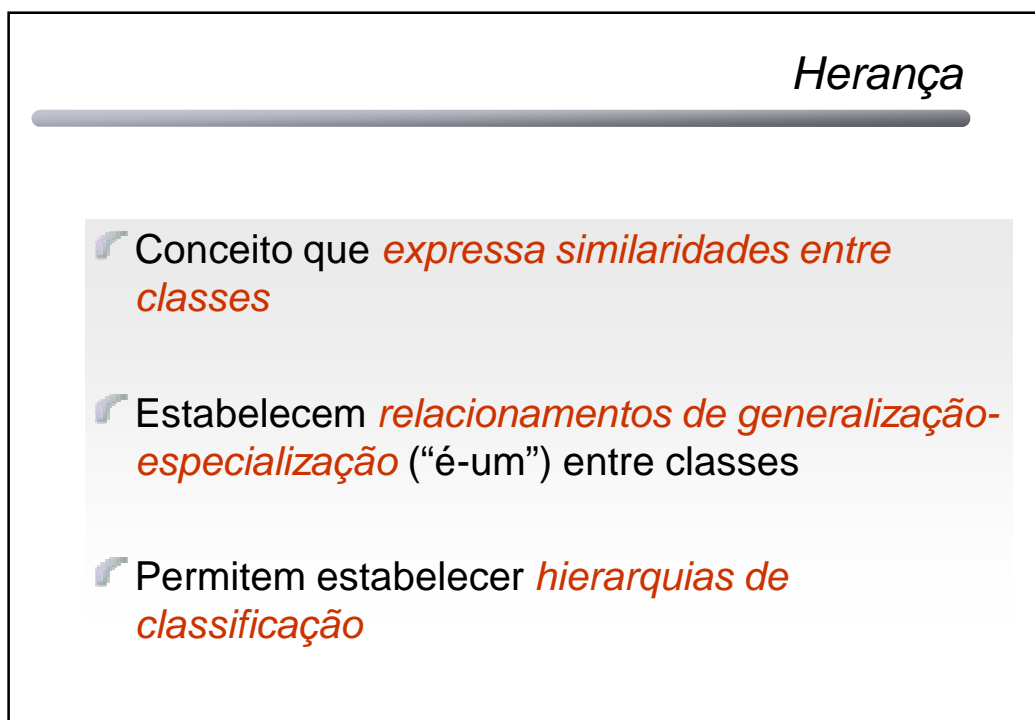
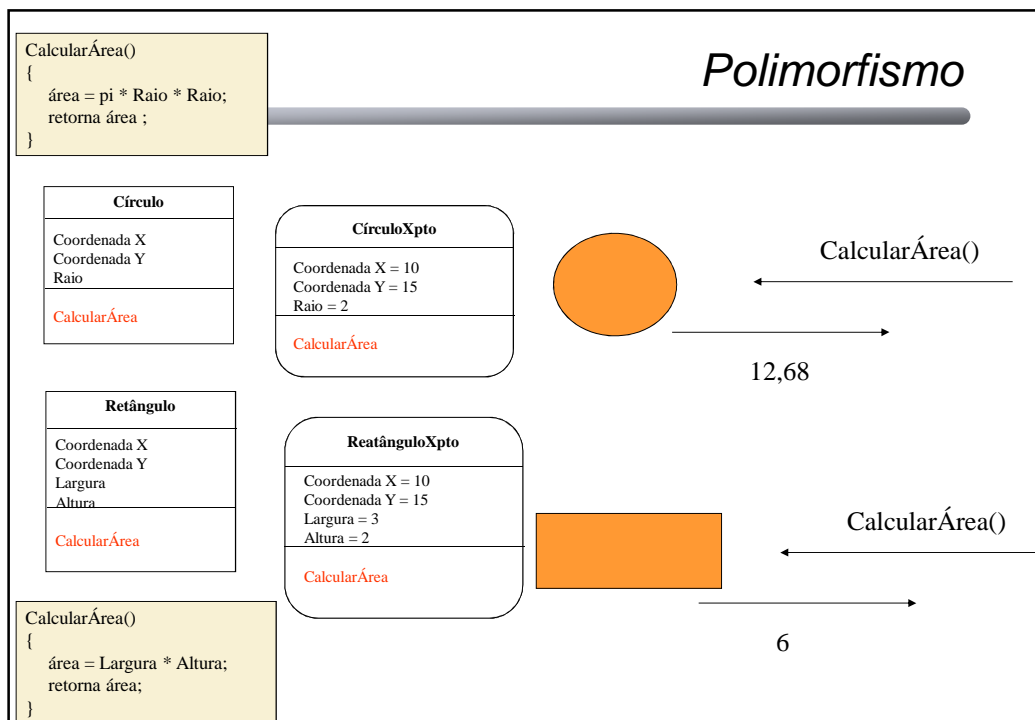
### Exemplo:

#### Funcionário Mário Sá

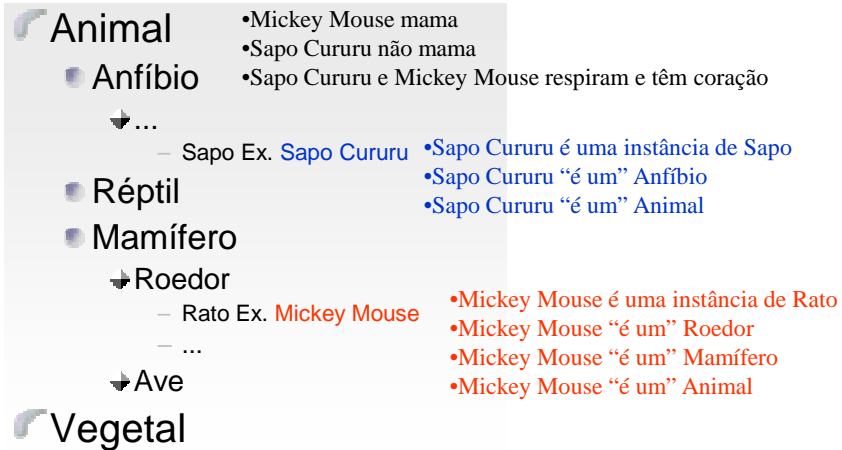


## Polimorfismo

- Possibilidade de enviar uma mesma mensagem para objetos de classes diferentes
- Cada uma das classes implementa um método específico para responder à mensagem
- Definição de protocolos comuns



## Herança



## Herança

Funcionário
Nome Nasc Salário
InformarSalário CalcularIdade



Funcionário Helena
Nome = Helena Maria Nasc = 09/03/1980 Salário = 4.000
InformarSalário CalcularIdade



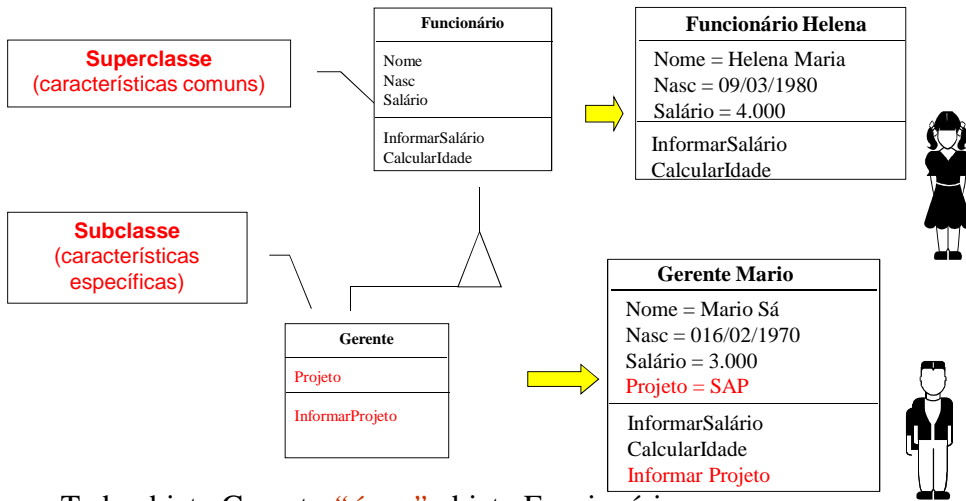
Gerente
Nome Nasc Salário <i>Projeto</i>
<i>InformarProjeto</i> InformarSalário CalcularIdade



Gerente Mario
Nome = Mario Sá Nasc = 016/02/1970 Salário = 3.000 <i>Projeto = SAP</i>
InformarSalário CalcularIdade <i>Informar Projeto</i>



## Herança

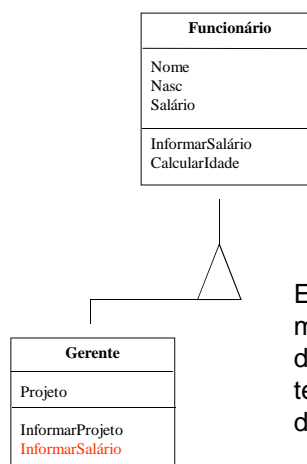


## Herança

### Polimorfismo

Suponha que gerentes recebam um adicional de 5% do salário por gerenciar projetos...

**InformarSalário:**  
retorna o valor contido no atributo 'Salário' adicionado com 5%



**InformarSalário:**  
retorna o valor contido no atributo 'Salário'

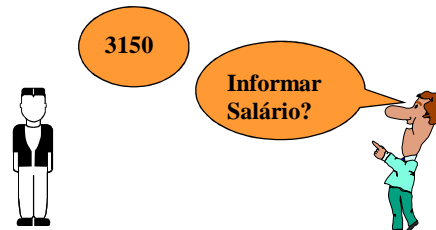
Embora o serviço tenha o mesmo nome, dependendo da classe, terá um comportamento distinto

## Herança

Funcionário Helena
Nome = Helena Maria
Nasc = 09/03/1980
Salário = 4.000
InformarSalário
CalcularIdade



Gerente Mario
Nome = Mario Sá
Nasc = 016/02/1970
Salário = 3.000
Projeto = SAP
InformarSalário
CalcularIdade
Informar Projeto



## Herança

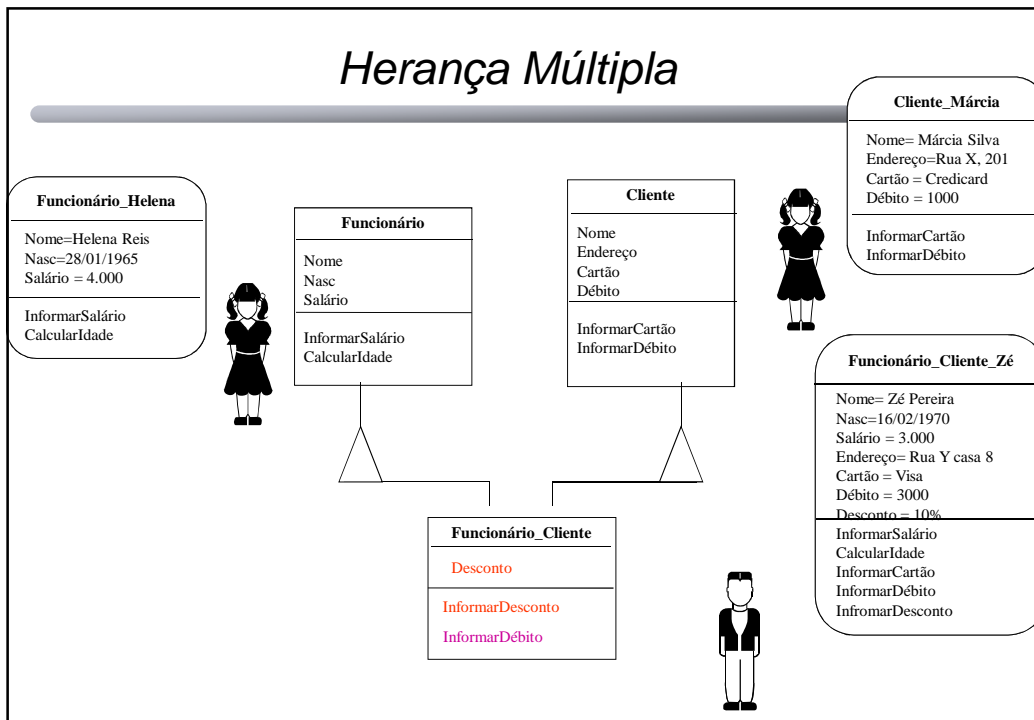
### Herança Múltipla

- Classes herdam características de várias classes

### Exemplo:

- Numa empresa administradora de cartões de crédito, alguns de seus **funcionários** são também seus **clientes**...

## Herança Múltipla

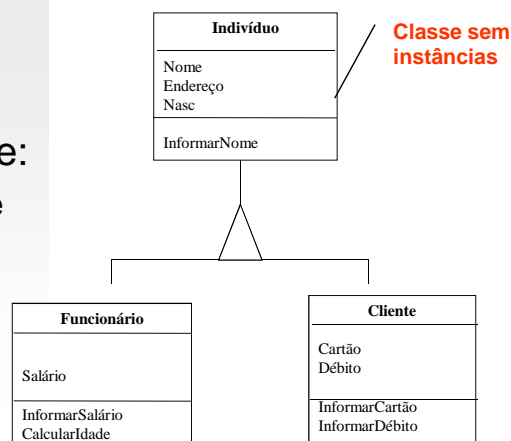


## Classe Abstrata

Representação de uma *classe que não possui instâncias*

Utilizada com objetivo de:

- diminuir a complexidade
- auxiliar a classificação e organização da modelagem

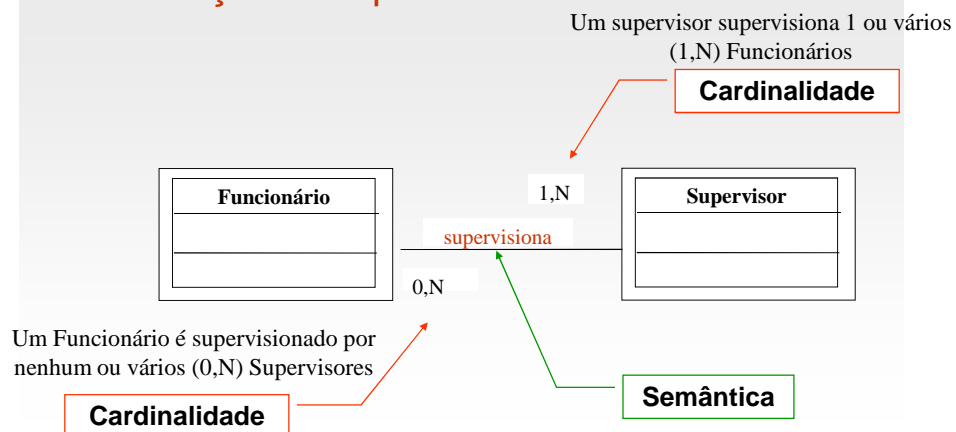


## Relacionamentos

- Instâncias de classes podem manter relacionamentos com instâncias de outras classes
- Associações simples*, com significado não definido previamente
- Associações com semântica pré-definida*

## Relacionamentos

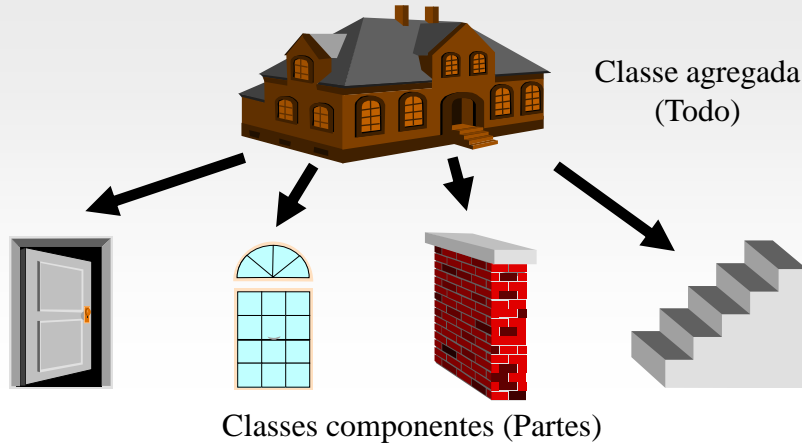
### Associações Simples





## Relacionamentos

### Agregação/Composição

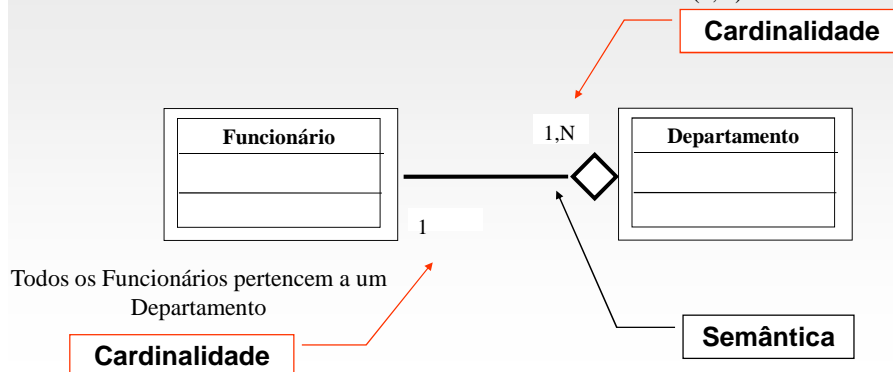


## Relacionamentos

### Agregação/composição

- Semântica pré-definida de composição

Um Departamento contém 1 ou vários (1,N) Funcionários



## *Cardinalidade*

---

☞ 0..1 - nenhum ou apenas um

☞ 0..N - nenhum ou vários

☞ 1 - somente um

☞ 1..N - vários

## *Métodos e Linguagens para OO*

---

**Um Processo (Espiral, RUP, ...)**

**+**

**UML - Unified Modeling Language**

