

Módulo V

Modelo Orientado a Objetos

Objetivos

- Consolidar conceitos básicos relacionados à Engenharia de Software
- Apresentar os principais conceitos do Modelo Orientado a Objetos

Créditos

Autor

Prof. Alessandro Cerqueira
[\(alessandro.cerqueira@hotmail.com\)](mailto:alessandro.cerqueira@hotmail.com)

Conceitos Introdutórios

Sistema

- **Sistema**

- Conjunto de elementos interdependentes que está inserido dentro de um ambiente e que é responsável por um processamento que recebe estímulos de entrada e produz uma saída específica.
- Não está ligado necessariamente à Computação



- **Ex:** Sistema Respiratório, Digestório, Circulatório, etc.

Conceitos Introdutórios

Dado e Informação

- **Dado**

- Representação de um **fato** em sua forma **primária**
 - *Observa-se o fato sem ou questionar sua essência ou razão.*
 - *Ex. Preço de produtos em um mercado, alunos em uma turma.*

- **Processamento**

- **Funcionalidades** previstas para o sistema de informação

- **Informação**

- Resultado do processamento de um conjunto de dados

Conceitos Introdutórios

Sistema de Informação

- **Sistema de Informação**



- O que é **Informação** em um Sistema, pode ser **Dado** para outro
 - Mudanças no **ambiente** ou no **processamento**, alteram o **sistema**.
 - **Valor do Cupom Fiscal** → **Informação** no Sistema de PDV e **Dado** no Sistema de Faturas de Cartão de Crédito

Conceitos Introdutórios

Processo de Software

- **Processo**

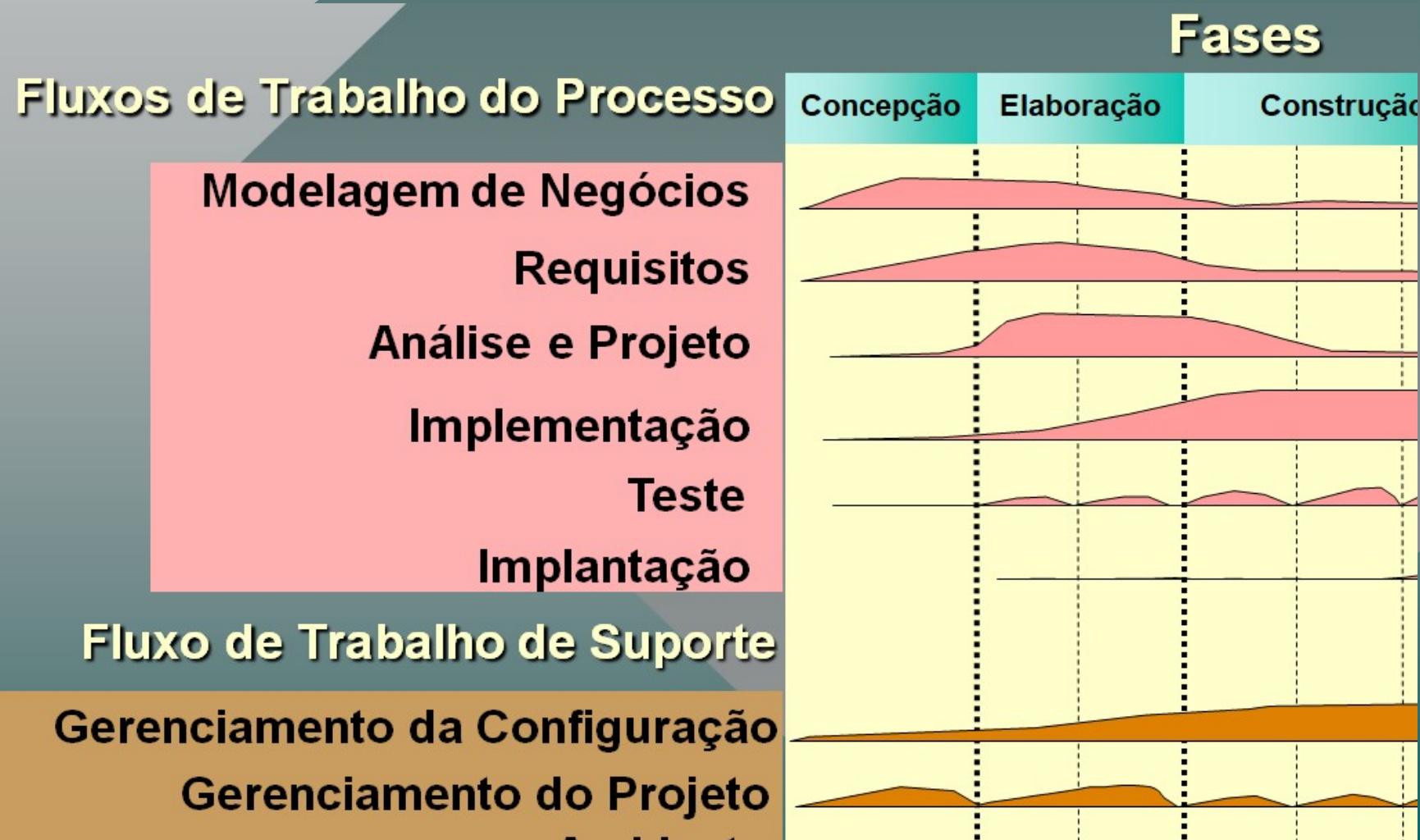
- Sequência contínua de **atividades**, **fatos** ou **operações** que apresentam certa unidade ou que se reproduzem com certa regularidade; andamento, desenvolvimento, marcha.
- Para se construir software é necessário estabelecer um **Plano de Ação**, indicando que **atividades** serão realizadas através do tempo
 - **Plano de Ação** → **Especificação do Processo de Software**
- Processo de Software ⇔ Processo de Desenvolvimento de Software
- As **atividades** do processo de software podem ser agrupadas em subconjuntos de acordo com os seus objetivos. Esses subconjuntos são chamados de **Fluxos de Trabalho**.
 - Um **Processo de Software** apresenta vários **Fluxos de Trabalho**;
 - Cada **Fluxo de Trabalho** apresenta várias **Atividades** a serem desempenhadas

Conceitos Introdutórios

Processo de Software e Fluxos de Trabalho

- Alguns **Fluxos de Trabalho** do Processo de Software:
 - **Levantamento de Requisitos**
 - **OBJETIVO:** Descobrir quais requisitos funcionais/não-funcionais o sistema deve apresentar
 - **Análise (de Requisitos)**
 - **OBJETIVO:** Especificar o software em **alto nível** através da construção de modelos (sem contemplar aspectos de limitação tecnológica)
 - **Projeto (Design)**
 - **OBJETIVO:** Especificar o software em **termos reais** e aceitáveis para o usuário/cliente. Aspectos considerados: Arquitetura do sistema, padrão de interface gráfica, linguagem de programação, gerenciador de BDs...
 - **Implementação**
 - **Teste**
 - **Implantação**
 - **Manutenção**
- Um dos principais instrumentos utilizados no processo de software é a **Abstração de Dados**.

Uma Visão do Processo de Software e de seus Fluxos de Trabalho através do Tempo e Esforço



Conceitos Introdutórios

Abstração de Dados

- Capacidade de representarmos conceitos do mundo real em sistemas computacionais na forma de dados ou de modelos de dados
 - Formalmente é uma descrição das *características essenciais* de uma *entidade* presente no *mundo real* através de uma *estrutura de dados*, que a distingue de todos os outros tipos de entidade e que proporciona *limites conceituais bem definidos*.
- Para podermos fazer abstração de dados é necessário que durante o *processo de software* utilizemos algum instrumento ou uma forma de pensar que nos capacite a produzir Tipos Abstratos de Dados

Conceitos Introdutórios

Abstração de Dados

- Ex:
 - Em Bancos de Dados Relacionais

```
Create table Departamento (
    id_dept numeric(4) not null,
    nome_dept varchar(40) not null,
    matr_gerente numeric(5) not null,
    primary key (id_dept),
    foreign key (matr_gerente) references Funcionário(id_func))
```

- Em Pascal

```
TURMA = REGISTER
```

```
    código : integer;
    nome   : String;
    prof    : ^Professor;
end;
```

Conceitos Introdutórios

Modelo de Dados

- Descreve um conjunto de conceitos e mecanismos para abstração de dados, nos auxiliando a representar domínios através de esquemas de dados.
 - Domínio: *Área ou assunto do mundo real para o qual construímos um sistema*
(ex: Instituições Financeiras, Universidade, Indústria)
- Principais Modelos de Dados
 - Modelo de Dados Relacional
 - Modelo de Dados Entidade-Relacionamento
 - Modelo de Dados Orientado a Objetos
- Um modelo de dados nos dá instrumentos para fazermos a abstração de dados gerando esquemas de dados para um determinado domínio.

Conceitos Introdutórios

Modelo de Dados (Revisão)

- Para ser um **Modelo de dados**, tem que apresentar um conjunto de **conceitos** e **mecanismos** para representação de um domínio.
- **Modelo Entidade-Relacionamento**
 - Conceitos: Entidade, Relacionamento, Atributo, Chave, Participação, Cardinalidade, Entidade Fraca, Especialização.
- **Modelo Relacional**
 - Conceitos: Relação (*Tabela*), Tupla (*Registro*), Atributo, Chave Primária, Chave Estrangeira, Restrição de Domínio, Restrição de Integridade Referencial

Conceitos Introdutórios

Esquema de Dados

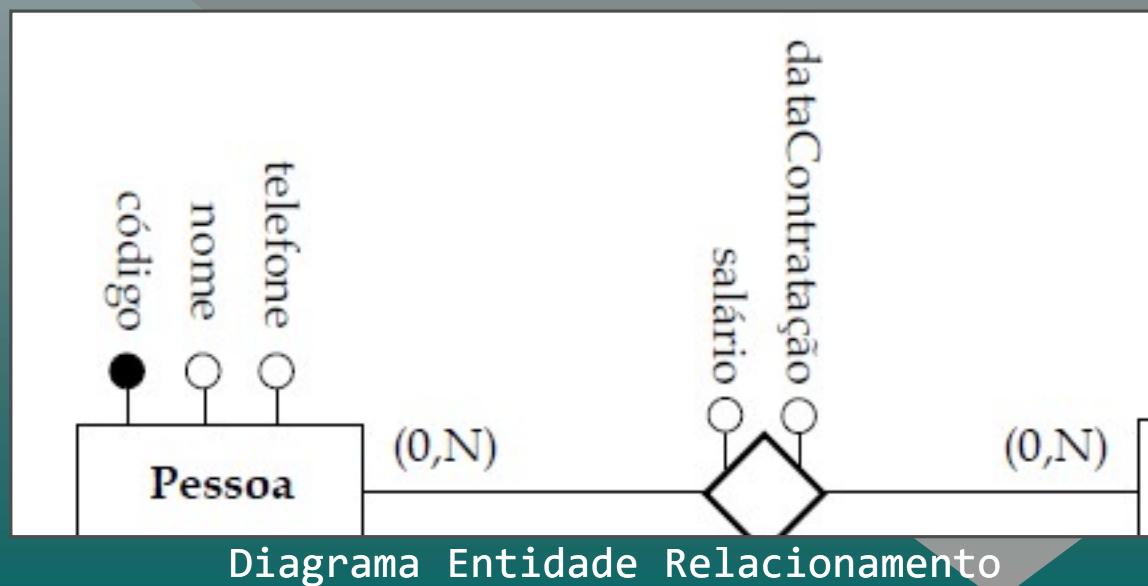
- É a **representação de um domínio específico** utilizando-se conceitos de um determinado modelo de dados.
 - Pode-se dizer que é o **produto do uso de um modelo de dados para algum domínio**
 - ex: Ao desenharmos um **Diagrama Entidade-Relacionamento**, estamos desenhandando um **Esquema de Dados** utilizando o **Modelo Entidade-Relacionamento**
- Pelo menos **três esquemas de dados** são produzidos durante o **processo de software**. Para associarmos cada um deles com o momento em que são produzidos, eles recebem os nomes de: **Modelo Conceitual, Modelo Lógico e Modelo Físico**.

Conceitos Introdutórios

Classificação dos Modelos de Dados

● **Modelo Conceitual**

- Representa o mundo real da forma como os usuários o percebem. Por isso são considerados modelos de **alto nível** de abstração.
- São produzidos durante as atividades de **Análise** (modelagem conceitual de dados)
- Modelos de Dados utilizados: Modelo Entidade-Relacionamento, Modelo Orientado a Objetos

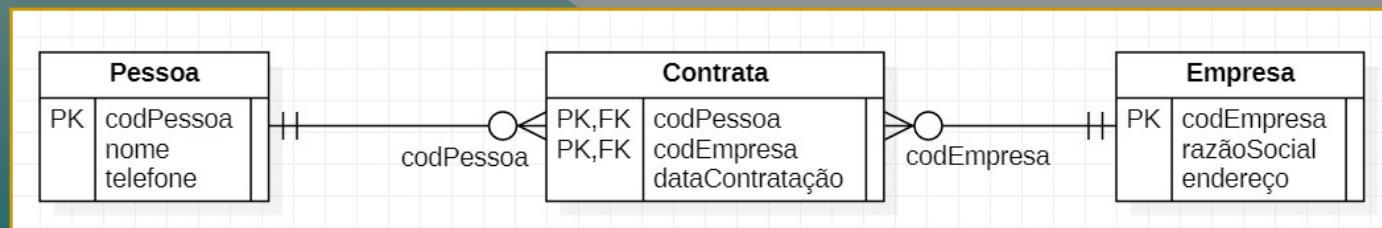


Conceitos Introdutórios

Classificação dos Modelos de Dados

- **Modelo Lógico (ou de Implementação):**

- Representa os dados sob alguma estrutura lógica utilizada pelos desenvolvedores em sua interação com os Sistemas Gerenciadores de Banco de Dados (SGBD).
- Apresentam algumas limitações para representar os fatos do mundo real na maneira como são percebidos, mas também não representam os dados na maneira como são fisicamente armazenados. Por isto são considerados modelos de **nível intermediário** de abstração.
- São produzidos durante as atividades do Fluxo de Trabalho de **Projeto (ou Design)**
- Modelos de Dados Utilizados: Modelo Relacional, Modelo Orientado a Objetos



Esquema Relacional expresso por Notação James Martin

Conceitos Introdutórios

Classificação dos Modelos de Dados

- **Modelo Lógico (ou de Implementação):**

```
Create Table Pessoa (
    codPessoa int not null,
    nome varchar(40) not null,
    telefone char(11) not null,
    constraint pk_pessoa primary key (codPessoa)
)
Create Table Empresa (
    codEmpresa int not null,
    razaoSocial varchar(60) not null,
    endereco varchar(80) not null,
    constraint pk_empresa primary key (codEmpresa)
)
Create Table Contrata (
    codPessoa int not null,
    codEmpresa int not null,
    dataContratacao Date not null,
    constraint pk_contrata primary key
        (codPessoa, codEmpresa),
    constraint fk_contrata_pessoa foreign key (codPessoa) references Pessoa(codPessoa),
    constraint fk_contrata_empresa foreign key (codEmpresa) references Empresa(codEmpresa)
)
```

Esquema Relacional Expresso com SQL

Conceitos Introdutórios

Classificação dos Modelos de Dados

- **Modelo Físico**

- Descrevem os detalhes de como os dados estão de fato armazenados, tais como: formato dos arquivos, caminho de acesso, ordem, etc. São considerados modelos de baixo nível.

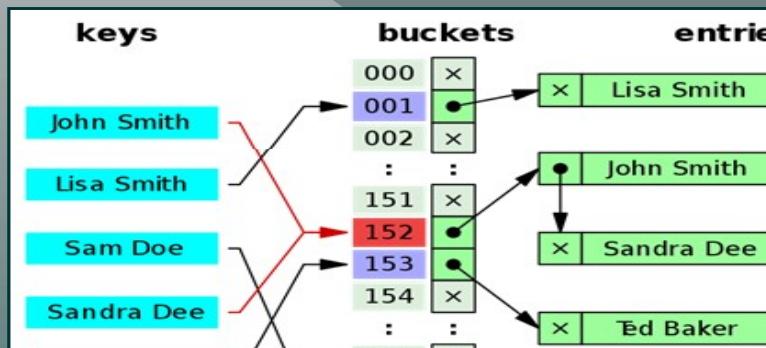
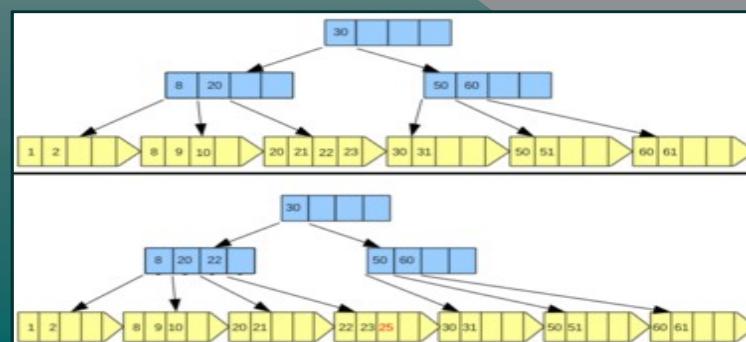
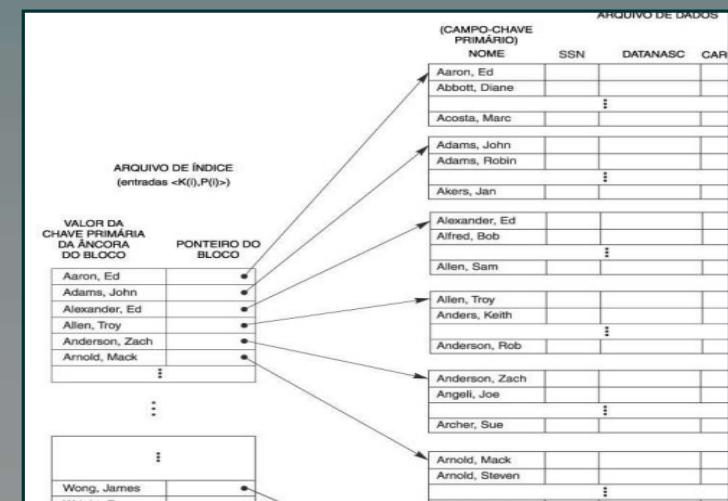


Tabela Hash



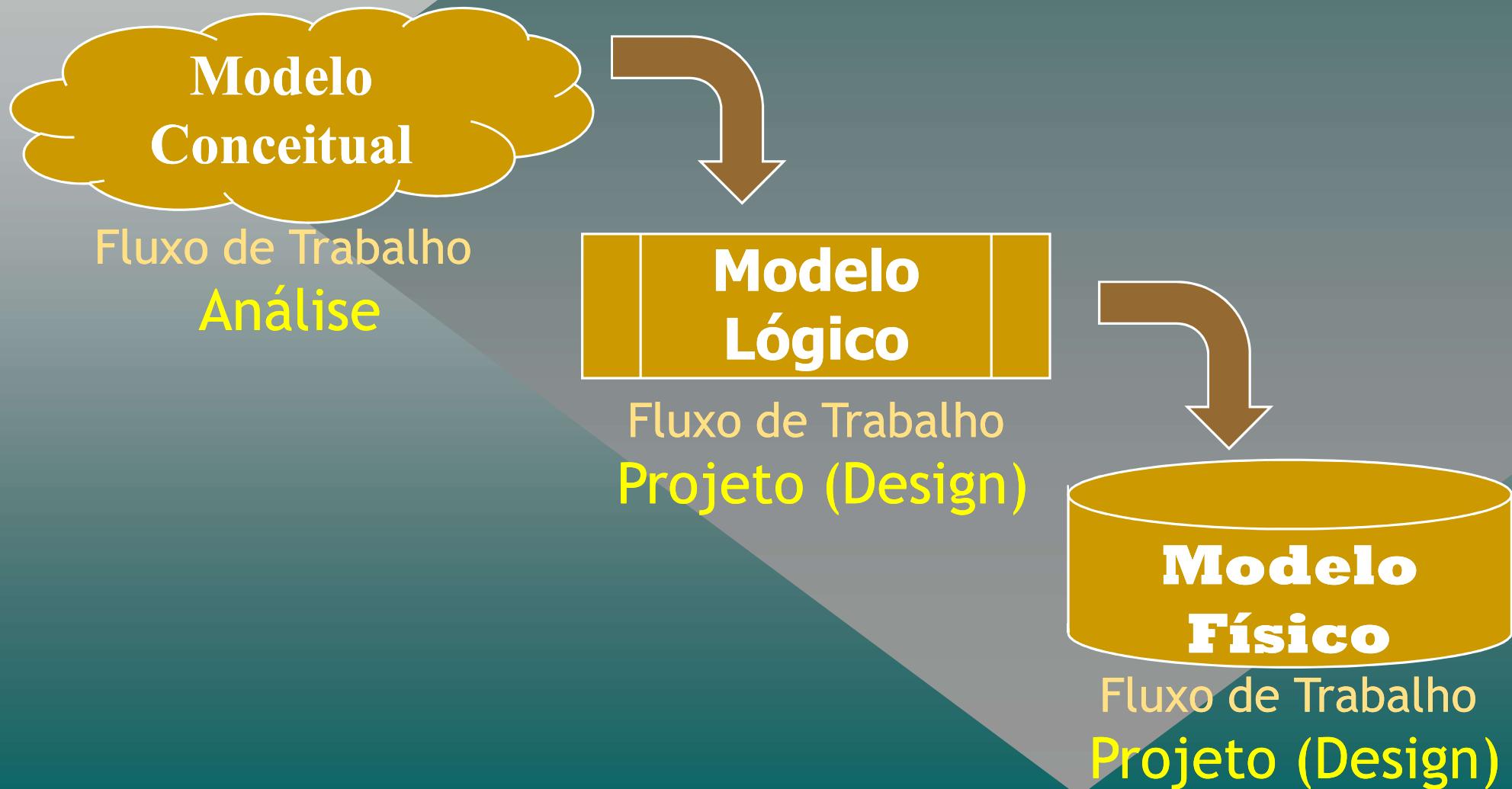
Árvore B



Índices

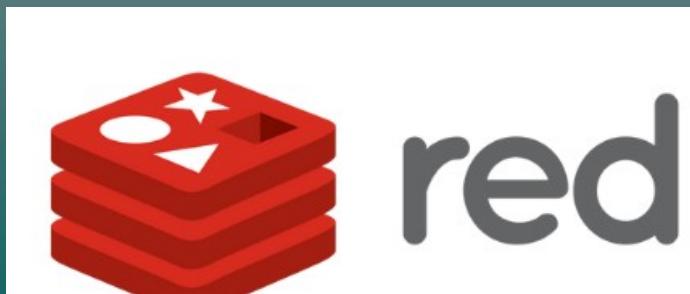
Conceitos Introdutórios

Geração dos Modelos através do Processo de Software



Bancos de Dados NOSQL

- Novas formas para geração de Modelos Lógicos
 - Banco de Dados Orientados a Colunas
 - Cassandra
 - Banco de Dados Orientados a Documentos
 - MongoDB
 - Banco de Dados Orientados a Grafos
 - Neo4J
 - Banco de Dados Chave-Valor
 - Redis



Conceitos Introdutórios

Orientação a Eventos X Orientação a Objetos

- Programação Orientada a Eventos
 - Estilo de programação que utiliza o seguinte paradigma:

SE OCORRER O EVENTO <X>
ENTÃO EXECUTE A CALLBACK (ou FUNÇÃO) < Y >

 - Amplamente utilizada na construção de interfaces gráficas
- Programação Orientada a Objetos
 - Baseada nos conceitos do modelo orientado a objetos.
- Em geral, os ambientes **RAD** (Rapid Application Development) e as **IDEs** (Integrated Development Environment) oferecem recursos que combinam as duas abordagens:
 - Ex: Eclipse, NetBeans, IntelliJ

Modelo Orientado a Objetos

Conceitos e Mecanismos

- Conceitos Estruturais
 - Classe, Objeto, Atributo, Encapsulamento (já vistos)
 - Polimorfismo
- Conceitos Comportamentais
 - Método, Mensagem (já vistos)
- Relacionamentos
 - Generalização/Especialização,
 - Agregação/Decomposição, Composição
 - Associação

Orientação a Objetos

Atributo, Método e Mensagem (Revisão)

- ☞ Todo **objeto** é instância de uma classe. Assim, a interface de um objeto é composta pelos atributos e operações definidas na interface de sua classe.
- Interface do Objeto ⇔ Interface da Classe
- ☞ **Atributo** é uma propriedade nomeada de um objeto que é capaz de armazenar um estado (valor).
- ☞ **Método** é uma implementação de uma operação que compõe a interface de um objeto.
- ☞ **Mensagem** é uma sinalização enviada para um objeto composta de nome e parâmetros. Ao receber a mensagem, o objeto executará o método que tiver o mesmo nome da mensagem.

Orientação a Objetos

Atributo, Método e Mensagem (Revisão)

- ⇒ Se uma **mensagem** “XYZ” é enviada a um objeto, este irá executar o método “XYZ” que está presente em sua interface.
- ⇒ O **conceito de método** é semelhante ao conceito de **função** da programação estruturada; porém **um método é sempre executado por um objeto**.
- ⇒ Assim, podemos nos abstrair com a idéia de que um objeto é uma entidade que possui um “processador” que é capaz de executar os métodos presentes em sua interface.
- ⇒ Para executar um método, o objeto pega o seu **código** na interface da classe e o executa. A descrição deste **código deve valer para todo e qualquer objeto da classe** em questão.
- ⇒ Assim como as **funções** em outras linguagens (ex. Linguagem C), o método **produz** um resultado ao final de sua execução. Dizemos que ele **retorna** (return) algo ao terminar de executar.
→ As exceções a esta regra são os métodos **void** e **construtores**

Orientação a Objetos

Atributo, Método e Mensagem

- Primeiro Exemplo

CLASSE PESSOA

Interface Privada

 nome : String
 dataNasc : Data

Interface Pública

informarIdade() : Inteiro

 Variáveis Locais

 Resultado : Inteiro

 Início

 Resultado := Ano(DataDeHoje) - Ano(**dataNasc do Objeto**);

 Se Mês(DataDeHoje) < Mês(**dataNasc do Objeto**)

 Retornar Resultado - 1;

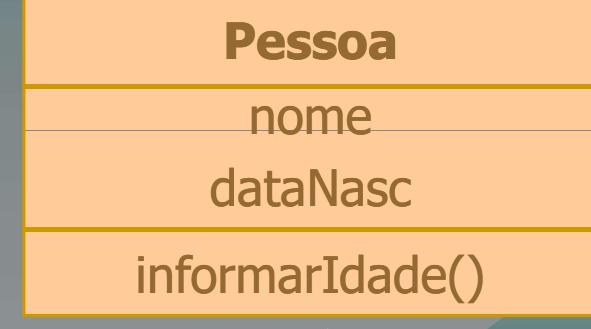
 Se Mês(DataDeHoje) = Mês(**dataNasc do Objeto**)

 Se Dia(DataDeHoje) < Dia(**dataNasc do Objeto**)

 Retornar Resultado - 1;

 Retornar Resultado;

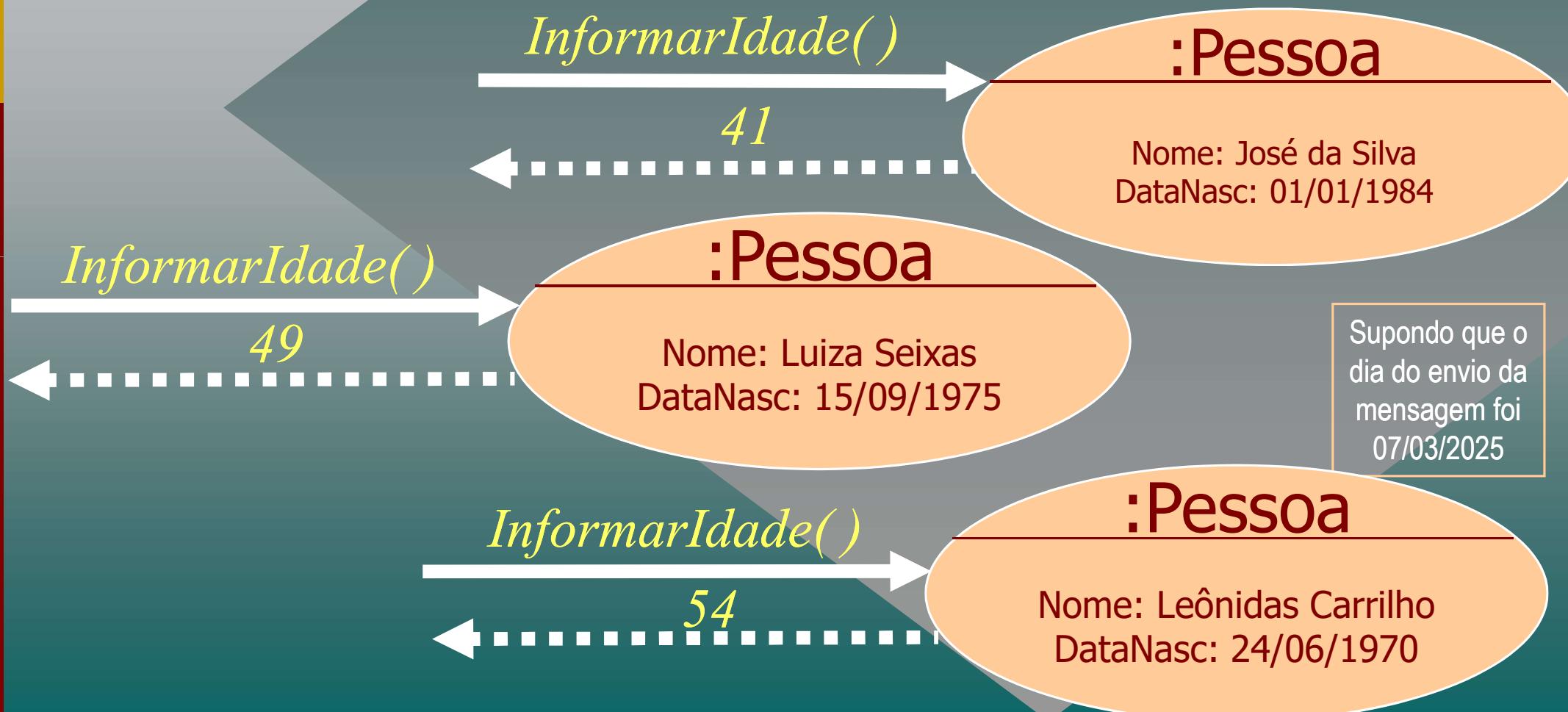
 Fim



Representação UML
da classe Pessoa

Orientação a Objetos

Atributo, Método e Mensagem



Orientação a Objetos

Atributo, Método e Mensagem

- Segundo Exemplo: Mensagem com Parâmetros

CLASSE PESSOA

Interface Privada

nome : String
dataNasc : Data

Interface Pública

informarNome() : String

Início

 Retornar nome do Objeto;

Fim

informarIdade(paramDia : Data) : Inteiro

Variáveis Locais

 resultado : Inteiro

Início

 resultado := Ano(paramDia) - Ano(dataNasc do Objeto);

 Se Mês(paramDia) < Mês(dataNasc do Objeto)

 Retornar Resultado - 1;

 Se Mês(paramDia) = Mês(dataNasc do Objeto)

 Se Dia(paramDia) < Dia(dataNasc do Objeto)

 Retornar Resultado - 1;

 Retornar Resultado;

Fim

Pessoa

 nome

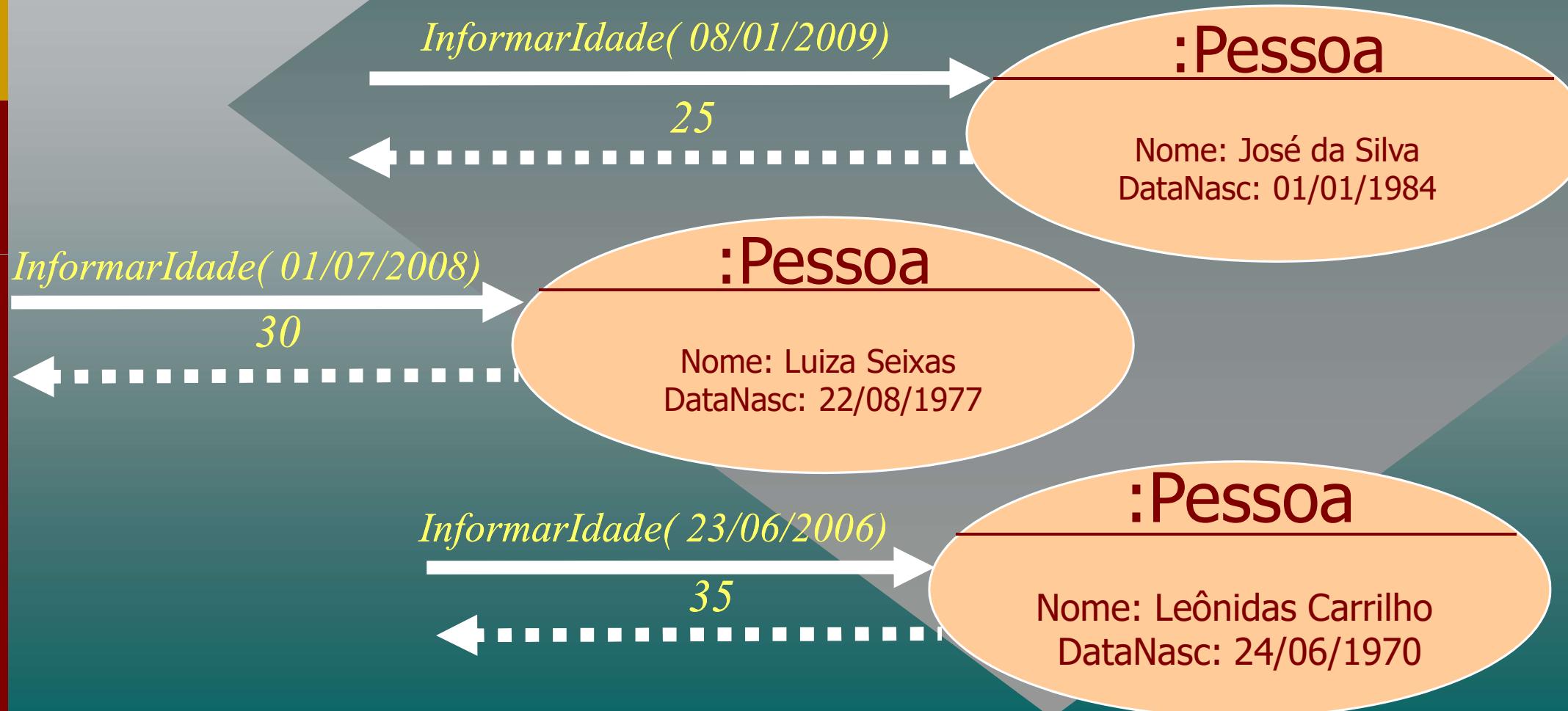
 dataNasc

 informarNome()

 informarIdade(Data)

Orientação a Objetos

Atributo, Método e Mensagem



Orientação a Objetos

Exercício

- Apresente três classes presentes no domínio “Universidade”.
 - Ex: Aluno, Professor, UnidadeDeEnsino
- Para cada classe, diga quais são os atributos presentes.
- Para cada classe, indique dois métodos.
- Para cada classe, desenhe dois objetos com seu estado.

Orientação a Objetos

Encapsulamento

→ Objetos são entidades que **encapsulam** (escondem) informação de seu estado (seus dados); ou seja, o estado de um objeto **não é acessível** a outros objetos a não ser através dos métodos pertencentes à interface pública do objeto (a parte da interface do objeto que é visível (conhecida) a outros objetos)



Orientação a Objetos

Interface Pública x Interface Privada

→ Propriedades da Classe

Conjunto de atributos e métodos que compõem uma classe.

Propriedade é um termo que qualifica tanto atributos quanto métodos.

→ Interface Pública

Conjunto de propriedades definidas em uma classe que não serão encapsuladas por suas instâncias. Assim, As propriedades pertencentes à interface pública de um objeto serão visíveis pelos objetos de outras classes.

→ Interface Privada

Conjunto de propriedades definidas em uma classe que serão encapsuladas por suas instâncias. Assim, as propriedades pertencentes à interface privada de um objeto não serão visíveis pelos objetos de outras classes.

Orientação a Objetos

Relacionamentos entre Classes

- Classes não existem sozinhas. No Modelo OO temos três tipos de relacionamentos entre classes:
 - Especialização/Generalização 
 - Agregação/Decomposição 
 - Associação 
- Os objetos estabelecem relacionamentos entre eles de acordo com os relacionamentos entre classes.
- Na UML também temos os relacionamentos (a serem vistos futuramente):
 - Realização 
 - Dependência 

Orientação a Objetos

Generalização/Especialização

- Tipo de relacionamento entre classes onde instâncias de uma categoria específica também consideradas instâncias de uma categoria mais abrangente.
- Utilizamos para derivar novas classes a partir de classes existentes através de um processo de refinamento.
- Prefira chamar somente de especialização.
- Sempre que estabelecemos uma especialização, entra em ação um mecanismo chamado de herança.



Pessoa é uma generalização de Aluno
Aluno é uma especialização de Pessoa

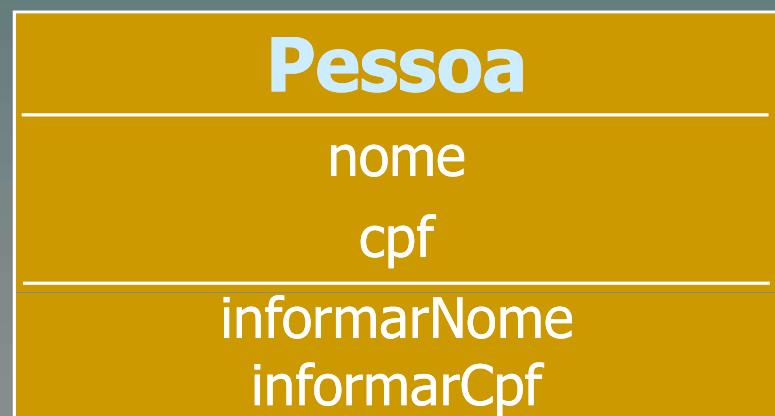
Orientação a Objetos

Generalização/Especialização

Nomenclatura

Generalização
ou Classe Base
ou Superclasse ou
Classe Mãe

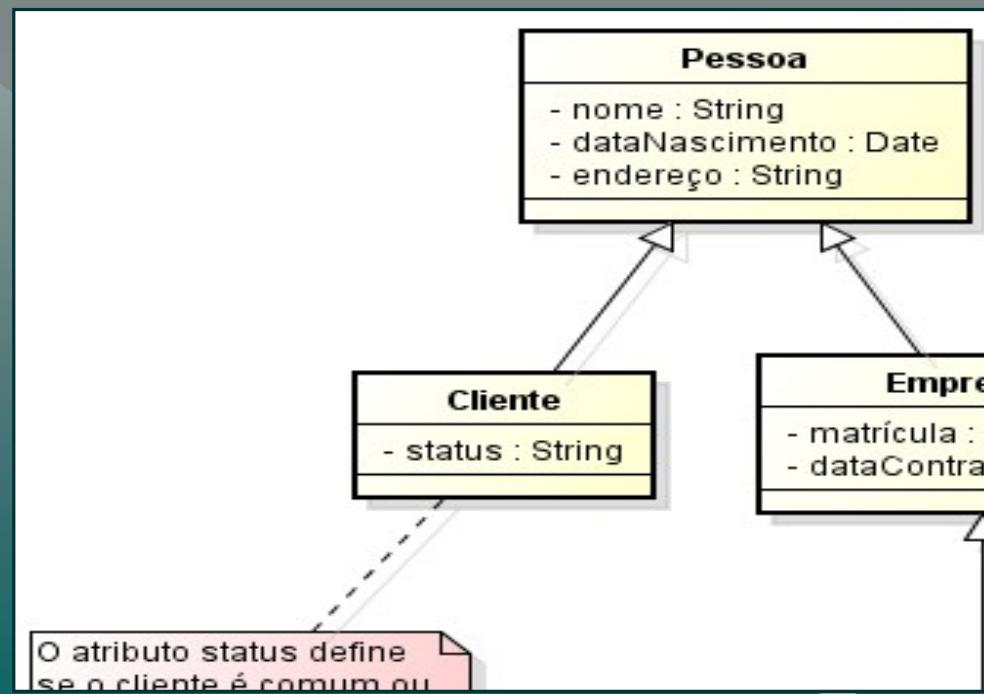
Especialização
ou Classe Derivada ou
Subclasse
ou Classe Filha



Orientação a Objetos

Herança

- **Transitividade:** Uma classe em uma hierarquia herda tanto as propriedades e relacionamentos de sua superclasse imediata quanto de suas superclasses não imediatas.



Orientação a Objetos

Herança

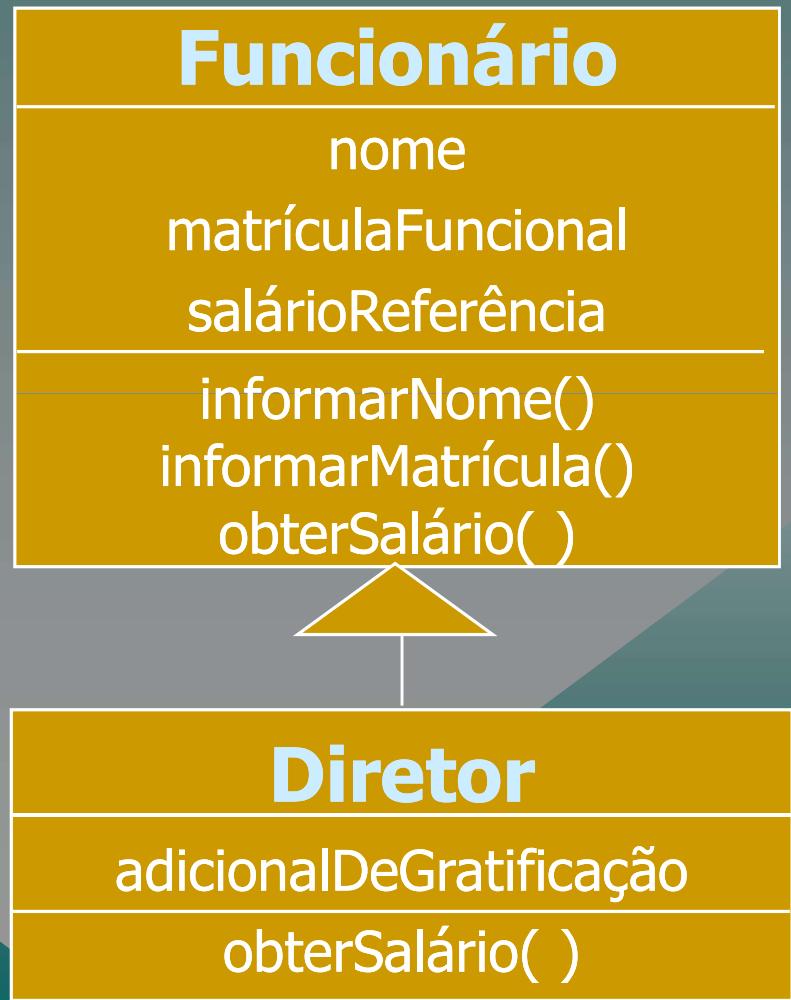
- Mecanismo que entra em ação *quando estabelecemos um relacionamento de especialização* que automaticamente *repassa* para a subclasse todas as propriedades presentes na sua superclasse.
- Uma classe derivada herda automaticamente os atributos e métodos sua classe base, mas pode seletivamente *adicionar novos métodos, adicionar novos atributos ou redefinir a implementação dos métodos herdados de sua classe base.*
 - A **redefinição** também é chamada de **sobrescrita, sobreposição** ou **overriding**.
- Considerando-se o exemplo do slide anterior, os atributos de Aluno são: nome, cpf e matrícula; e seus métodos são: informarNome(), informarCpf() e informarMatrícula()

Orientação a Objetos

Herança – Redefinição de Métodos

- Exemplo:

- Suponha que em um domínio, **Diretor** é sempre considerado como um **Funcionário**.
- Suponha também que o **salário** de um **Funcionário** é simplesmente o seu **salário de referência**.
- Suponha que o **salário** de um **Diretor** é o seu **salário de referência mais** o seu **adicional de gratificação**.



Orientação a Objetos

Herança – Redefinição de Métodos

CLASSE FUNCIONÁRIO

Interface Privada

nome : String
 matrículaFuncional : String
 salárioReferência : Real

Interface Pública

informarNome() : String
 Início
 Retornar nome do Objeto;
 Fim

informarMatrícula() : String
 Início
 Retornar matrícula do Objeto;
 Fim

obterSalário() : Real
 Início
 Retornar salárioReferência do Objeto;
 Fim

CLASSE DIRETOR

ESPECIALIZAÇÃO DE FUNCIONÁRIO

Interface Privada

adicionalDeGratificação : Real

Interface Pública

obterSalário() : Real
 Início
 Retornar salárioReferência do Objeto +
 adicionalDeGratificação do Objeto;
 Fim

Redefinição do Método
ObterSalário()
 herdado de **Funcionário**



Orientação a Objetos

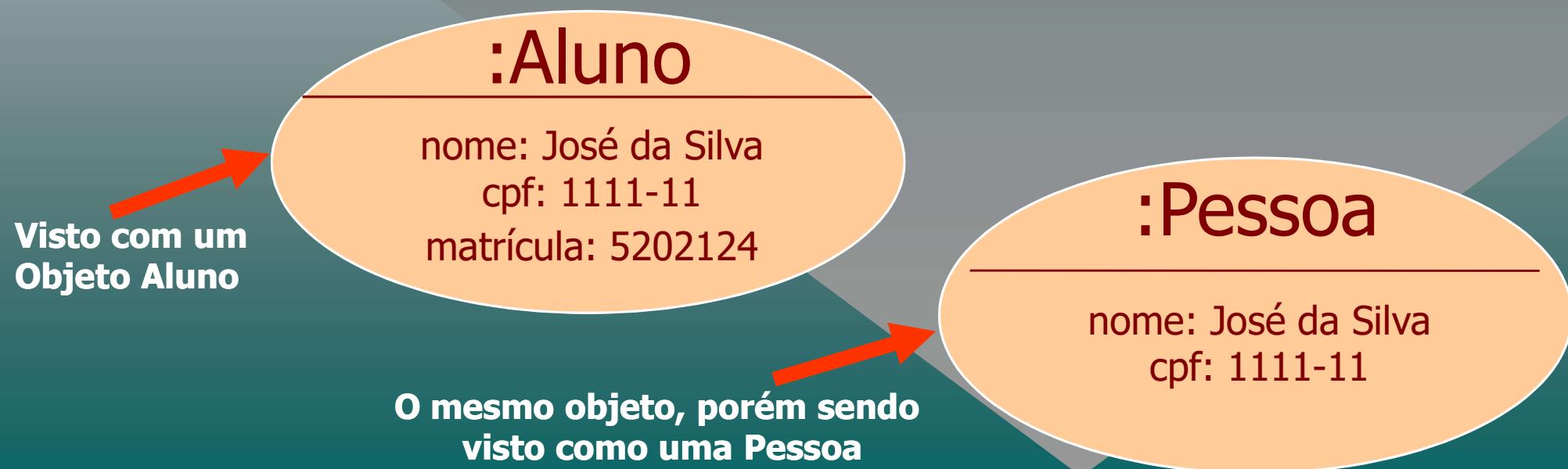
Polimorfismo

- ⇒ Capacidade que os objetos de uma subclasse têm de serem percebidos (**vistos**) pelas demais entidades na forma de objetos da sua superclasse.
- ⇒ Só ocorre nas instâncias de uma subclasse
- ⇒ Se uma classe “X” é especialização de “Y”, então podemos dizer um objeto “X” é tanto instância da classe “X” quanto é instância da classe “Y”.

Orientação a Objetos

Exemplo de Polimorfismo

- O objeto não muda de forma! O que muda de forma é a maneira que as demais entidades o percebem em determinadas situações.
- Ex: Um Aluno é visto dentro da universidade como Aluno; entretanto ao andar na rua os demais indivíduos (que não sabem que ele é aluno) o percebem como Pessoa.



Orientação a Objetos

Sobrecarga / Overload

- Quanto temos dois ou mais métodos com o mesmo nome, porém com *assinaturas* diferentes.
- A assinatura de um método é determinada pela estrutura de sua lista de parâmetros.
- Exemplo:

`somar(a : inteiro, b : inteiro) : inteiro`

`somar(a : real, b : real) : real`

`somar(a : inteiro, b : real) : real`

`somar(a : complexo, b : complexo) : complexo`

`somar(a : real, b : inteiro) : real`

Orientação a Objetos

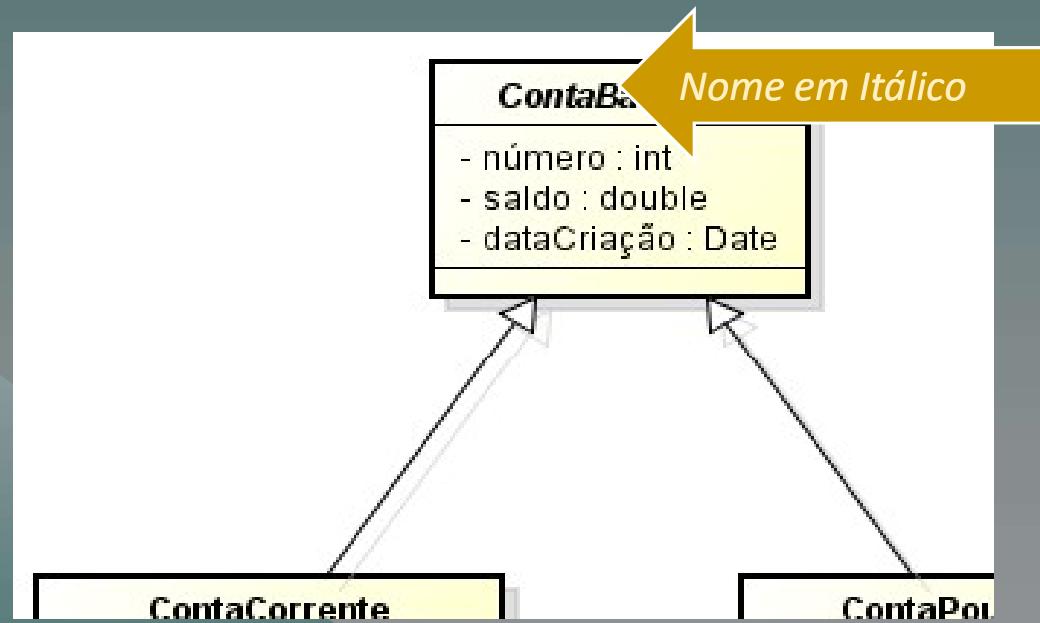
Classe Abstrata

- Também chamada de *não-instanciável*.
- É a classe que **não possui e nem possuirá instâncias próprias**.
 - Um **objeto** é uma instância própria de uma **classe** quando este objeto **não puder ser visto como um** instância de uma subclasse.
- Projetamos classes abstratas para **facilitar a derivação** de novas classes através do estabelecimento de especializações.
- Se em algum instante observarmos um objeto de uma classe abstrata, na realidade esta instância é pertencente a uma especialização da classe abstrata.

Orientação a Objetos

Classe Abstrata

- Exemplo:



- A determinação de uma classe ser abstrata ou não dependerá do **domínio** ou do **sistema** a ser implementado.

Orientação a Objetos

Herança Múltipla

→ É quando uma classe é especialização direta de mais de uma classe.



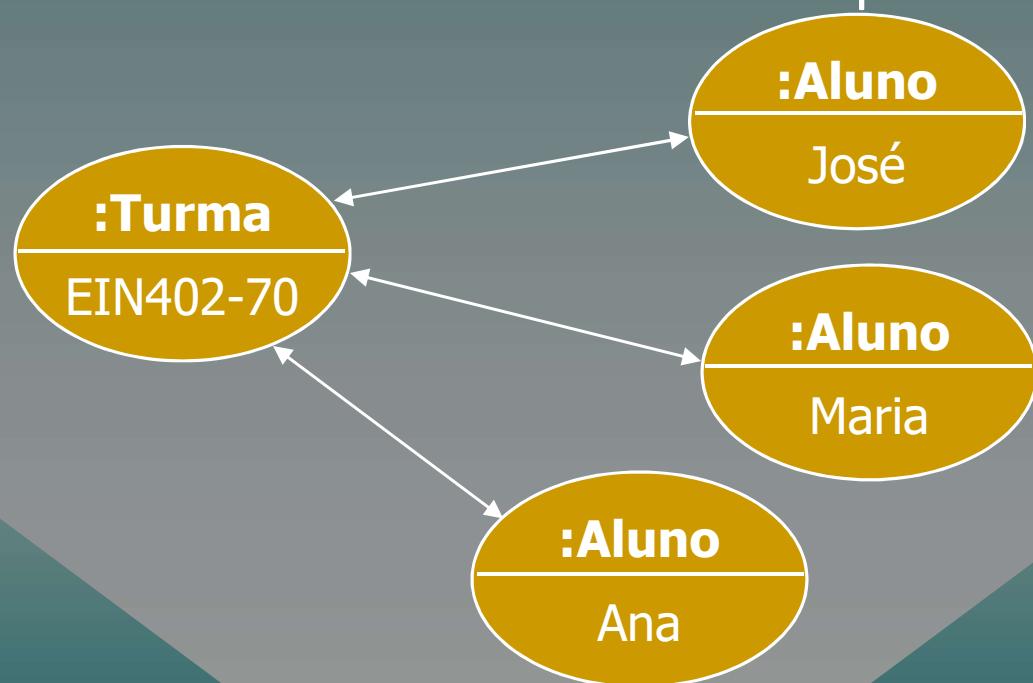
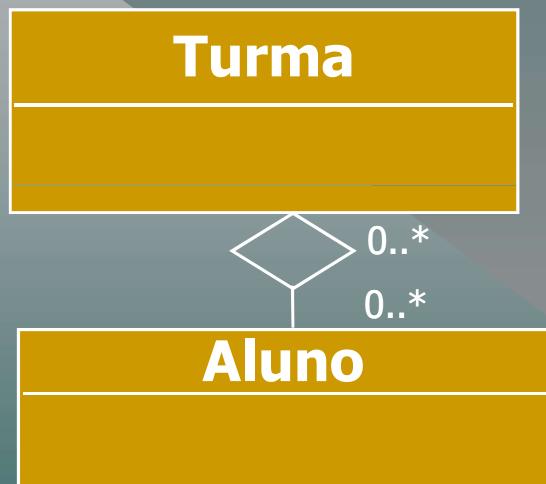
→ As linguagens mais novas **não** dão mais suporte à herança múltipla

Não é Herança Múltipla!!!

Orientação a Objetos

Agregação/Decomposição

→ Tipo de relacionamento onde instâncias de uma classe são compostas por outras.



Turma é uma agregação de Alunos e Aluno é uma decomposição de Turma.

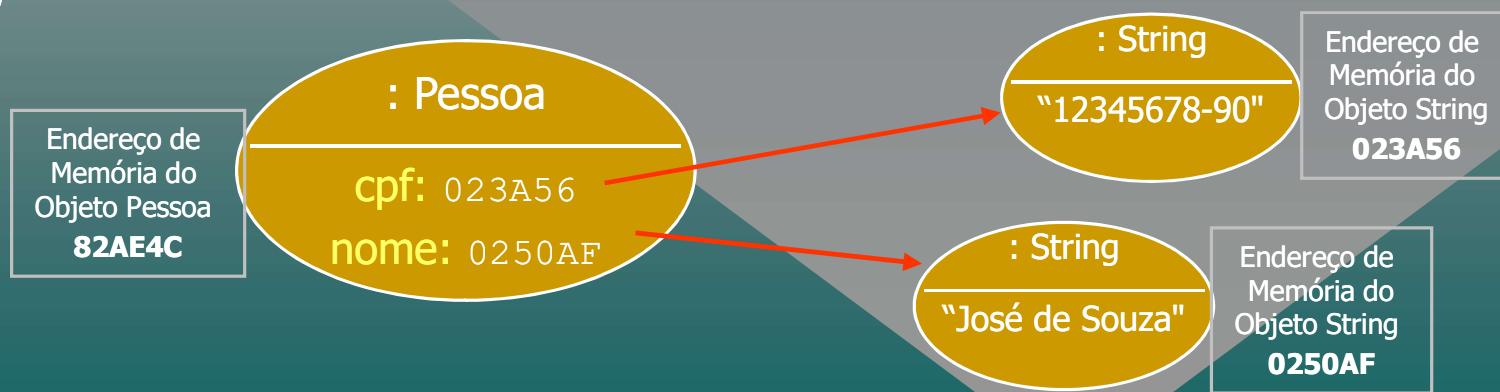
Uma Turma é composta de Um ou Vários Alunos e um Aluno compõe Zero ou Várias Turmas

Orientação a Objetos

Agregação/Decomposição

- **Vínculo**

- Expressa uma ligação de um objeto com outra instância.
 - Semelhante à idéia de *Chave Estrangeira* do *Modelo Relacional*
 - Ligação ou referência direta
- É baseado em um relacionamento de Agregação e Associação.
- Nas linguagens O.O., vínculos são implementados com o uso de **ponteiros**
 - Ponteiro → número que expressa um endereço de memória onde está um valor ou objeto.



Agregação e Composição

- A UML apresenta diferenças entre os conceitos Agregação (losango não-hachurado)
 - Quando a destruição da instância agregadora não implica na destruição automática das instâncias agregadas
 - Exemplo: Em uma universidade, a remoção dos dados de uma turma não implica na remoção dos dados dos alunos associados a ela.
 - **Composição** (losango hachurado)
 - Quando a destruição da instância agregadora implica na destruição automática das instâncias agregadas
 - Exemplo: Se uma universidade deixar de existir, os seus departamentos também deixarão de existir.



Orientação a Objetos

Associação

- Expressa um relacionamento entre classes que possui determinado significado (**semântica**)
- Em cada associação devemos especificar sua semântica (e seu sentido de leitura), e a multiplicidade (participação e cardinalidade).

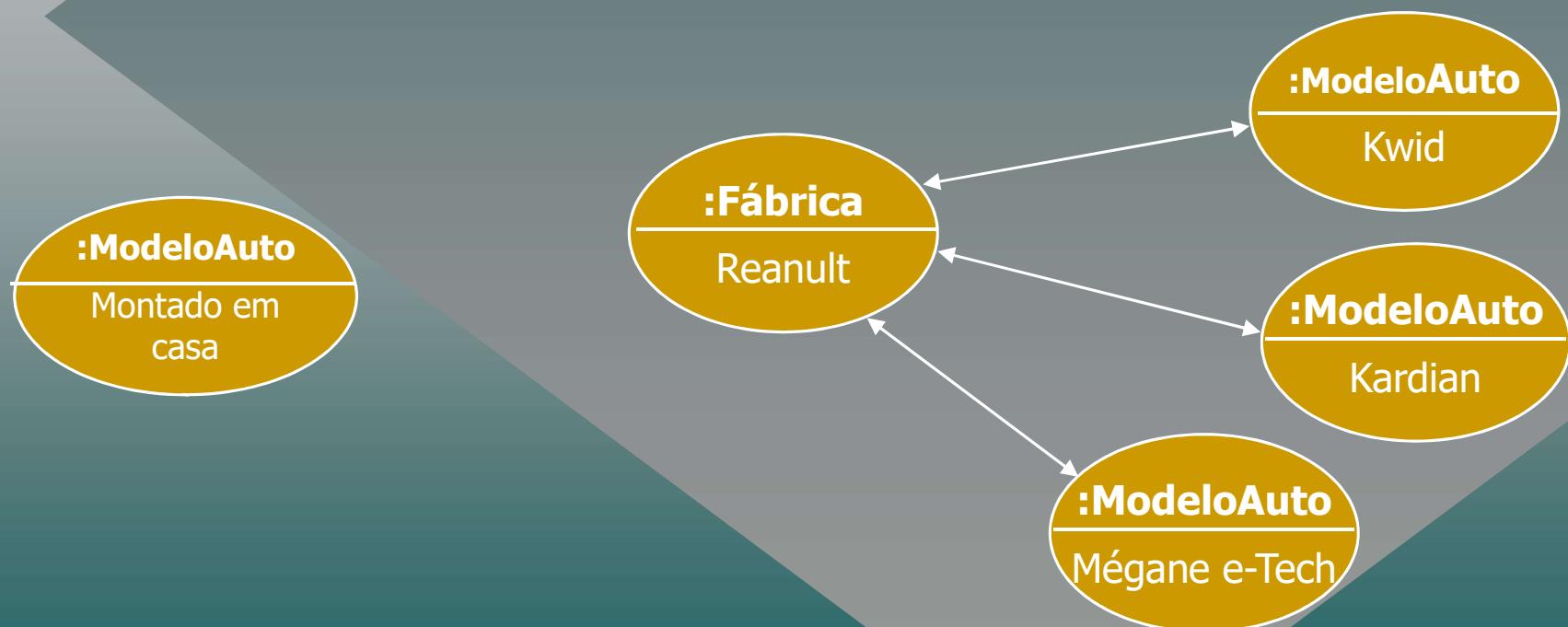


Um modelo de automóvel é **fabricado por** zero ou uma fábrica.
Uma fábrica **fabrica** um ou vários modelos de automóveis.

Orientação a Objetos

Associação

- Ex:



Orientação a Objetos

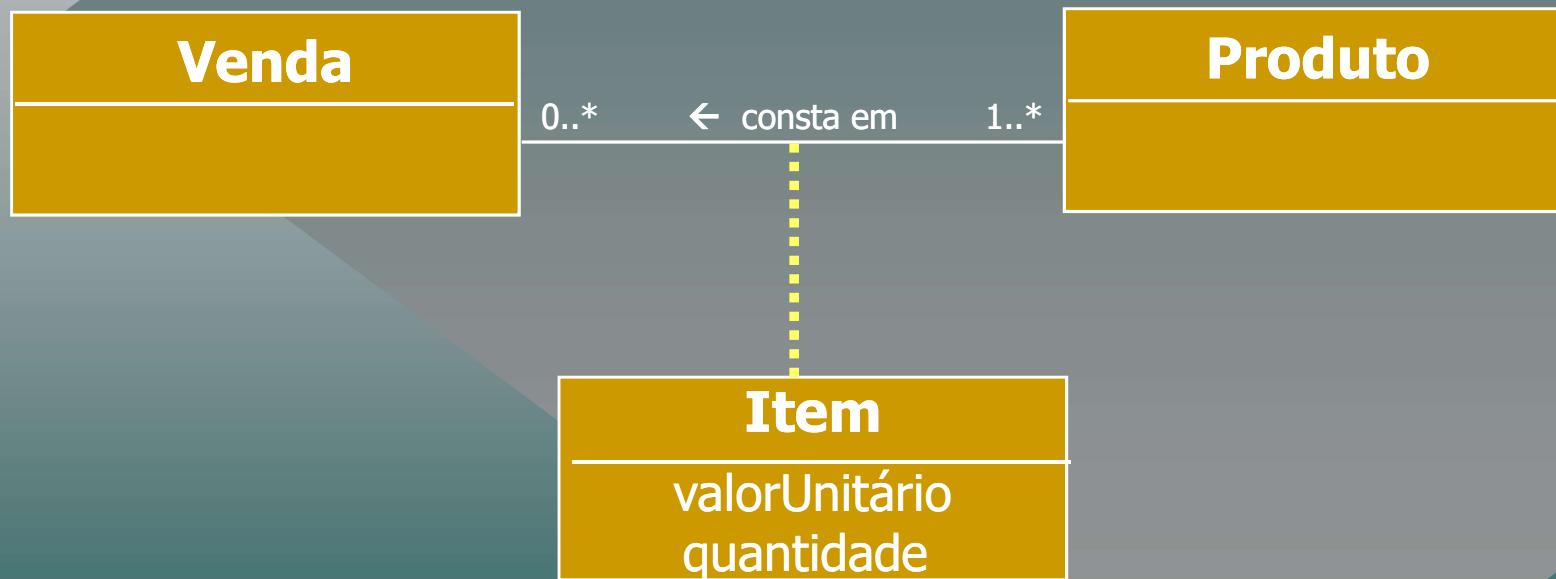
Classe Associativa

- **Classe Associativa (ou Classe de Associação)**
 - Definimos classes de associação quando desejamos vincular propriedades no estabelecimento de agregações ou associações.
 - Semelhante ao conceito de relacionamento com atributos definido no Modelo E-R
 - Toda vez que o relacionamento for estabelecido entre objetos, teremos vinculado ao relacionamento as propriedades definidas na classe de associação.

Orientação a Objetos

Classe Associativa

- Exemplo:



Toda vez que um objeto **Produto** estiver associado a uma **Venda**, deverá ficar “pendurado” nessa associação um objeto **Item** que indica o valor unitário do produto na venda, bem como a quantidade de itens do produto na venda.

Orientação a Objetos

Quando não Representar Classe Associativa

- Exemplo:



Só devemos ter Classe Associativa se **o par de instâncias não se repetir**. Se puder ocorrer a repetição, fazemos a modelagem usando duas associações. No exemplo, a Empresa pode ter realizado a contratação do mesmo Trabalhador mais de uma vez.

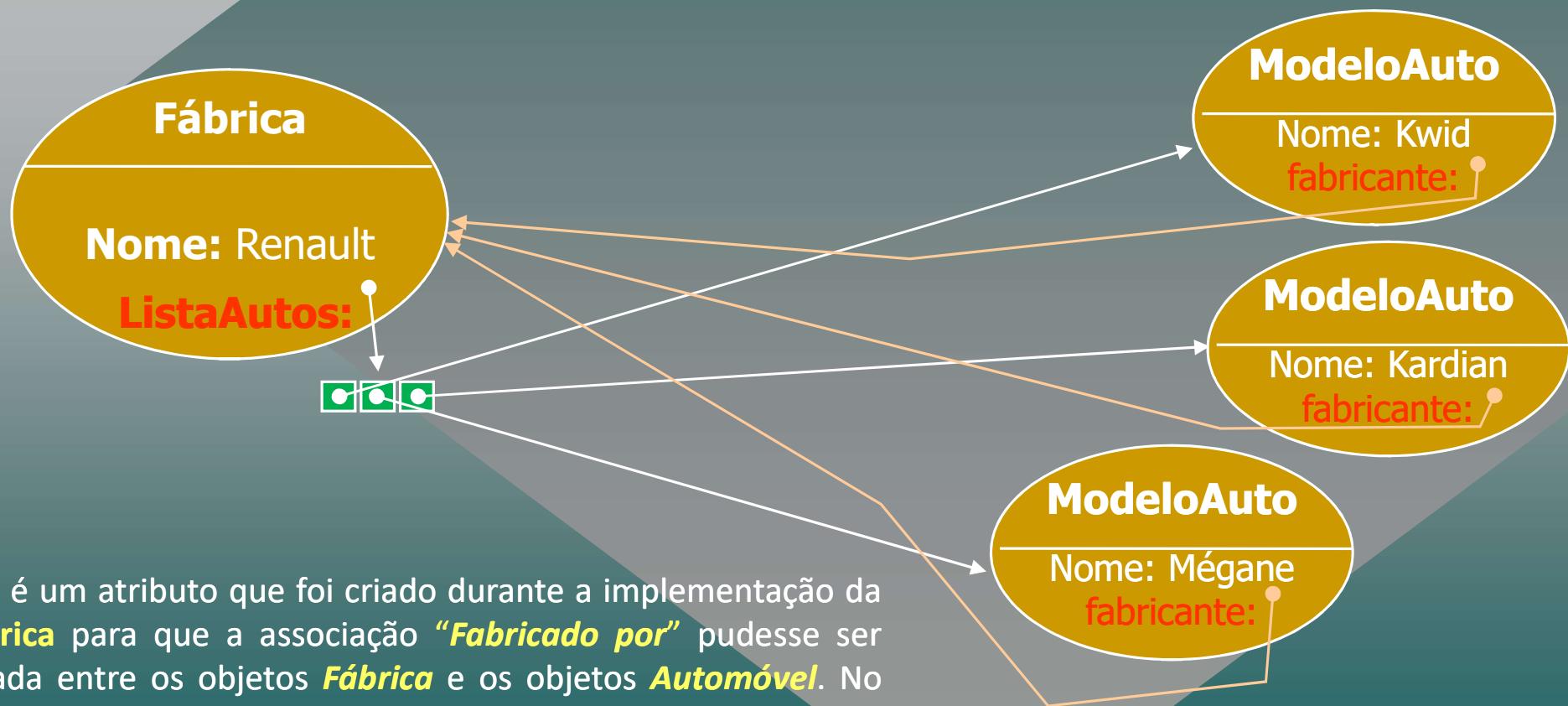
Orientação a Objetos

Implementação dos Relacionamentos nas Linguagens OO

- No caso de Generalização/Especialização, a materialização ocorre automaticamente em cada objeto especializado, pois este é tanto instância da classe generalizada quanto da classe especializada.
- Já nas Agregações quanto das Associações, a materialização não ocorre automaticamente.
 - É necessário que os objetos das classes participantes do relacionamento estabeleçam formalmente **vínculos** indicando que determinada agregação ou associação ocorre entre eles.
 - No caso das Linguagens OO, é comum **criar atributos** para estabelecermos referências para os objetos que participam do relacionamento.

Orientação a Objetos

Materialização dos Relacionamentos



ListaAutos é um atributo que foi criado durante a implementação da classe **Fábrica** para que a associação “**Fabricado por**” pudesse ser materializada entre os objetos **Fábrica** e os objetos **Automóvel**. No sentido inverso, foi adicionado o atributo **fabricante** em **ModeloAuto** para referenciar a **Fábrica**.

Orientação a Objetos

Materialização dos Relacionamentos

- **Vínculos Unidirecionais e Bidirecionais**
 - Observe que no exemplo proposto anteriormente, tanto a instância Fábrica aponta (conhece) para os ModeloAutos, quanto o ModeloAuto aponta para a Fábrica. Assim esta associação é **Bidirecional**. Se implementássemos somente em um dos lados, a associação seria **Unidirecional**.
- **Comparação com o Modelo Relacional**
 - A criação de atributos não é algo do Modelo OO. Ao adicionarmos uma **chave estrangeira** em uma tabela também estamos estabelecer um relacionamento entre tuplas.
 - A única **diferença** é que no Modelo Relacional o relacionamento estabelecido com **chave estrangeira** é **naturalmente bidirecional** (apesar de colocarmos o atributo somente em um dos lados).

Orientação a Objetos

Especificação dos Relacionamentos

- Uma vez descobertas as classes de um domínio, devemos especificar seus relacionamentos. Toda classe deve participar pelo menos de um relacionamento.
 - Será **especialização** se pudermos sempre afirmar que “**TODO XXX É UM YYY**”
 - *Todo Aluno é uma Pessoa*
 - Será **agregação** se pudermos sempre afirmar que “**TODO XXX É COMPOSTO DE YYY**”
 - *Toda Turma é composta de Alunos*
 - Se não for nenhum dos casos acima, descobrir a semântica para especificar a **associação**

Orientação a Objetos

Especificação dos Relacionamentos

- Lembre-se que a agregação nada mais é que uma associação com a semântica “é composto de”.
- Há situações em que um relacionamento pode ser representado semanticamente como uma agregação ou como uma associação. Isto **depende do ponto de vista do analista ou dos especialistas do domínio**.
 - Ex: Aluno e Turma
 - *Turma é composta de Alunos (Agregação); ou*
 - *Aluno está matriculado em Turma (Associação)*

Orientação a Objetos

Multiplicidade, Participação e Cardinalidade

- **Participação**
 - Indica se toda instância de uma classe **deve** ou **não** participar da agregação ou associação.
 - Valores: 0 ou 1.
 - Sugere-se indicar a participação obrigatória se, *no momento da criação do objeto*, a instância deverá participar do relacionamento.
- **Cardinalidade**
 - Indica o **número máximo de vezes** que um objeto **participa** de uma agregação ou associação.
 - Valores: 1 ou *
- Na UML a **Multiplicidade** é a indicação destes dois conceitos