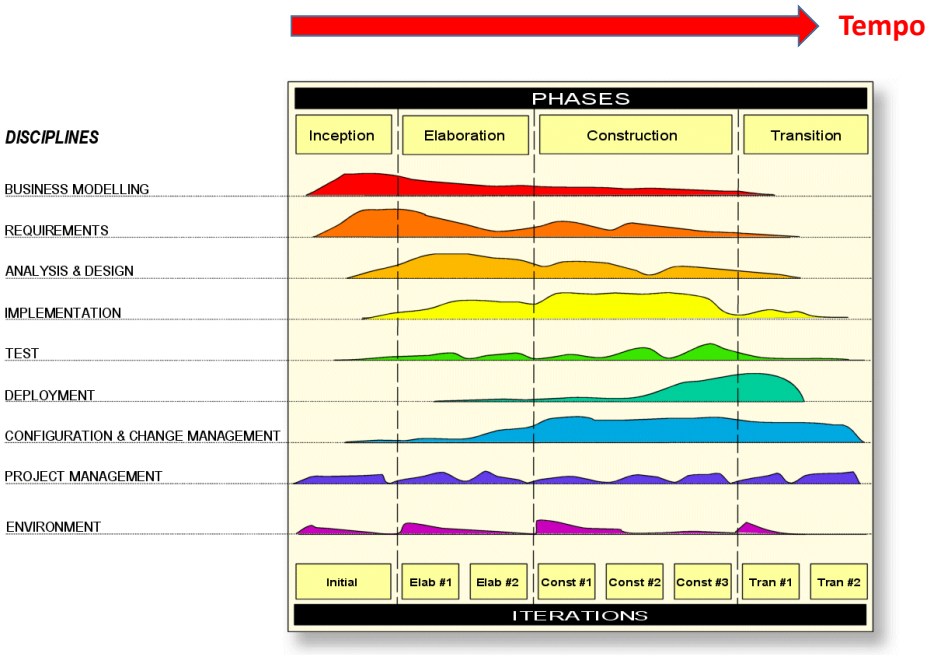
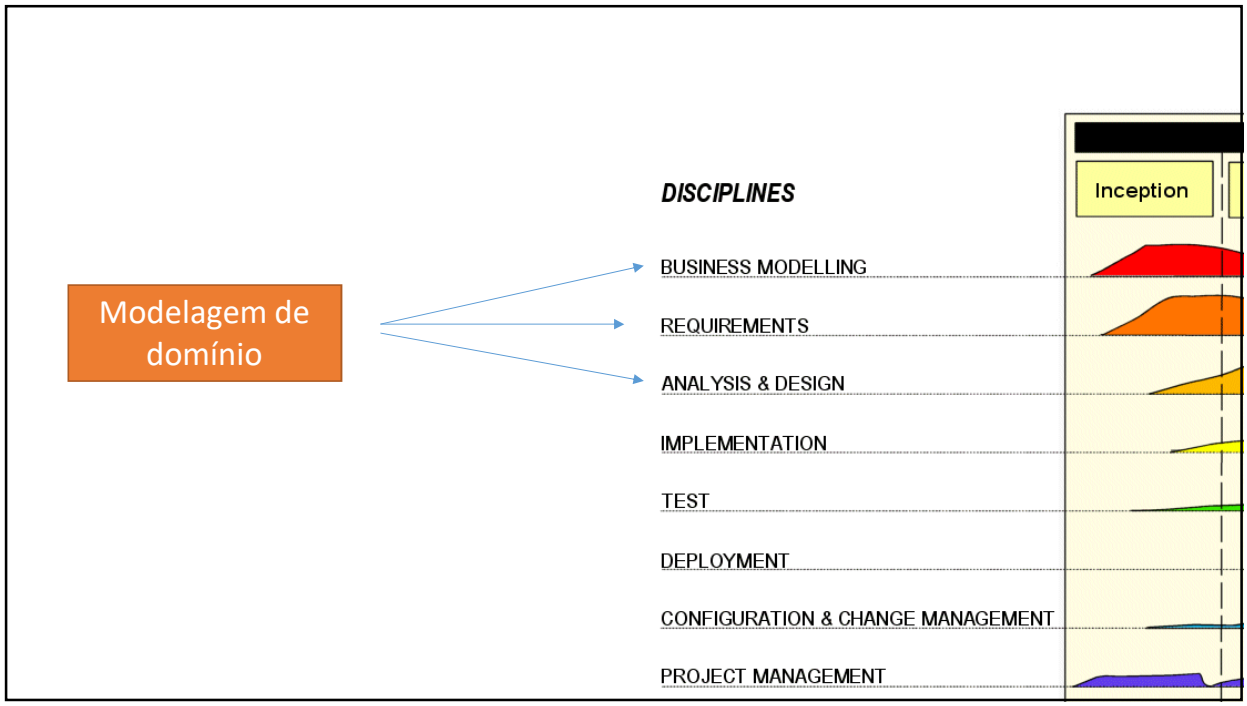
Entendendo modelagem de domínio e modelagem conceitual

Curso: Modelagem Conceitual com Diagrama de Classes da UML
https://www.udemy.com/user/nelio-alves
Prof. Dr. Nelio Alves





O que é modelo de domínio?

Domíno: é a área de negócio observada

Modelo de domínio: é um modelo que descreve

- As entidades do domínio
- As inter-relações entre elas

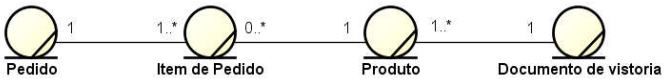
Móveis Angular
Avenida Vale do Silício, 751, Bairro Estudante
Uberlândia-MG

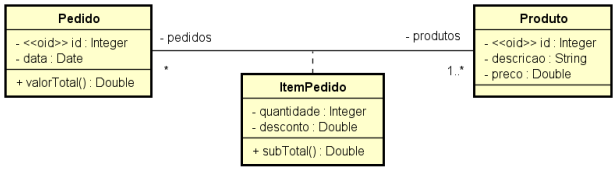
NOTA FISCAL

Pedido n°:	1001
Data	25/06/2017

Detalhes do pedido:

Produto	Descrição	Preço unitário	Quantidade	Desconto	Subtotal
8021	Cadeira simples	400.00	4	0%	1600.00
8055	Mesa retangular	1500.00	1	10%	1350.00
8014	Estante	2000.00	1	5%	1900.00
				Total do pedido:	4850.00





Níveis de abstração do modelo de domínio

Nível	Responsável	Objetivo
Conceitual ou de Análise (de negócio)	Analista de negócio	Descrever as entidades do domínio **(do negócio)** e suas inter-relações: **Independentemente de SISTEMA**
Conceitual ou de Análise (de sistema)	Analista de sistemas	Descrever as entidades do domínio (**do sistema**) e suas inter-relações: • **Independentemente de PARADIGMA E TECNOLOGIA**
Lógico ou de Design	Projetista	Descrever as entidades do domínio (**do sistema**) e suas inter-relações: • **Preso a um PARADIGMA (ex: relacional, orientado a objetos)** • **Independentemente de TECNOLOGIA**
Físico ou de implementação	Implementador	Descrever as entidades do domínio (**do sistema**) e suas inter-relações: • **Preso a um PARADIGMA (ex: relacional, orientado a objetos)** • **Preso a uma TECNOLOGIA (ex: Java, C#, PHP, Python, Ruby, NodeJS)**

Nível Conceitual ou de Análise (de negócio)

Móveis Angular
Avenida Vale do Silício, 751, Bairro Estudante
Uberlândia-MG

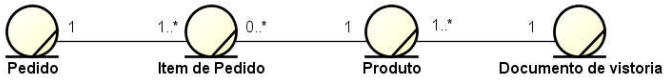
NOTA FISCAL

Pedido nº:	1001
Data	25/06/2017

Detalhes do pedido:

Produto	Descrição	Preço unitário	Quantidade	Desconto	Subtotal
8021	Cadeira simples	400.00	4	0%	1600.00
8055	Mesa retangular	1500.00	1	10%	1350.00
8014	Estante	2000.00	1	5%	1900.00

Total do pedido: 4850.00



Nível Conceitual ou de Análise (de sistema)

Móveis Angular
Avenida Vale do Silício, 751, Bairro Estudante
Uberlândia-MG

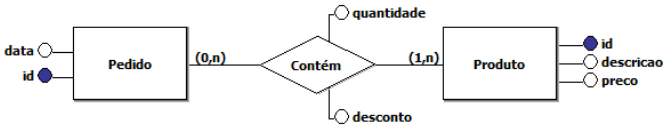
NOTA FISCAL

Pedido nº:	1001
Data	25/06/2017

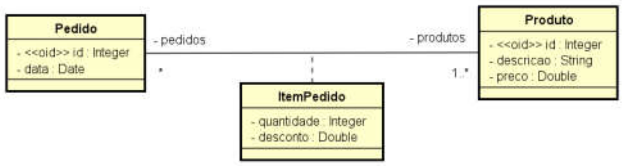
Detalhes do pedido:

Produto	Descrição	Preço unitário	Quantidade	Desconto	Subtotal
8021	Cadeira simples	400.00	4	0%	1600.00
8055	Mesa retangular	1500.00	1	10%	1350.00
8014	Estante	2000.00	1	5%	1900.00

Total do pedido: 4850.00



OU:



Nível Lógico ou de Design

Preso ao paradigma: **relacional**

Móveis Angular
Avenida Vale do Silício, 751, Bairro Estudante
Uberlândia-MG

NOTA FISCAL

Pedido nº:	1001
Data	25/06/2017

Detalhes do pedido:

Produto	Descrição	Preço unitário	Quantidade	Desconto	Subtotal
8021	Cadeira simples	400.00	4	0%	1600.00
8055	Mesa retangular	1500.00	1	10%	1350.00
8014	Estante	2000.00	1	5%	1900.00

Total do pedido: 4850.00

```
Pedido(id, data)

Produto(id, descricao, preco)

ItemPedido(id_Pedido, id_Produto, quantidade, desconto)
    id_Pedido references Pedido(id)
    id_Produto references Produto(id)
```

OU:

Nível Lógico ou de Design

Móveis Angular
Avenida Vale do Silício, 751, Bairro Estudante
Uberlândia-MG

NOTA FISCAL

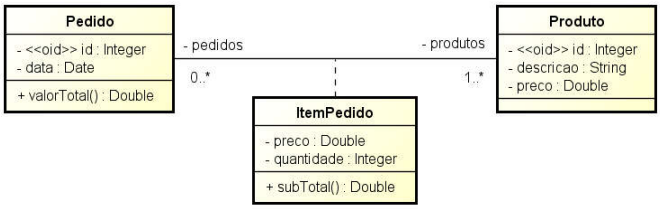
Pedido nº:	1001
Data	25/06/2017

Detalhes do pedido:

Produto	Descrição	Preço unitário	Quantidade	Desconto	Subtotal
8021	Cadeira simples	400.00	4	0%	1600.00
8055	Mesa retangular	1500.00	1	10%	1350.00
8014	Estante	2000.00	1	5%	1900.00

Total do pedido: 4850.00

Preso ao paradigma: **orientado a objetos**



Nível Físico ou de Implementação

Móveis Angular
Avenida Vale do Silício, 751, Bairro Estudante
Uberlândia-MG

NOTA FISCAL

Pedido nº:	1001
Data	25/06/2017

Detalhes do pedido:

Produto	Descrição	Preço unitário	Quantidade	Desconto	Subtotal
8021	Cadeira simples	400.00	4	0%	1600.00
8055	Mesa retangular	1500.00	1	10%	1350.00
8014	Estante	2000.00	1	5%	1900.00

Total do pedido: 4850.00

Preso ao paradigma: **relacional**
Preso à tecnologia: **MySQL (dialeto SQL)**

```
CREATE TABLE `produto` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `descricao` varchar(255) DEFAULT NULL,
  `preco` float DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;


(...)
```

Nível Físico ou de Implementação

Móveis Angular
Avenida Vale do Silício, 751, Bairro Estudante
Uberlândia-MG

NOTA FISCAL

Pedido n°:	1001
Data	25/06/2017

Detalhes do pedido:

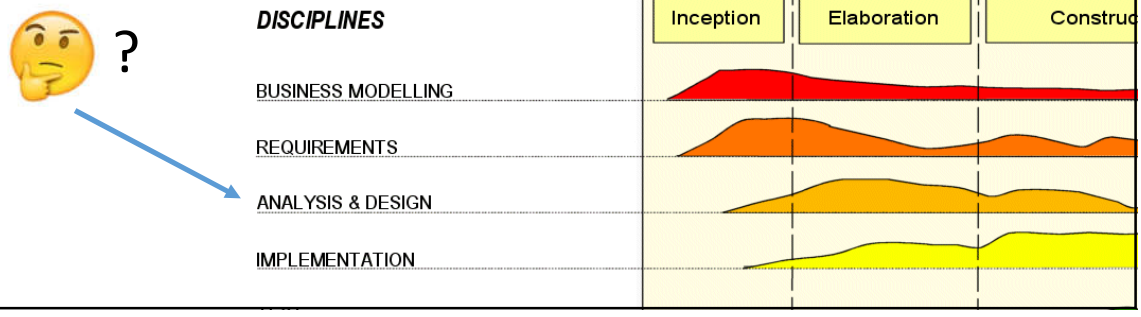
Produto	Descrição	Preço unitário	Quantidade	Desconto	Subtotal
8021	Cadeira simples	400.00	4	0%	1600.00
8055	Mesa retangular	1500.00	1	10%	1350.00
8014	Estante	2000.00	1	5%	1900.00
				Total do pedido:	4850.00

Preso ao paradigma: **orientado a objetos**
Preso à tecnologia: **Java**

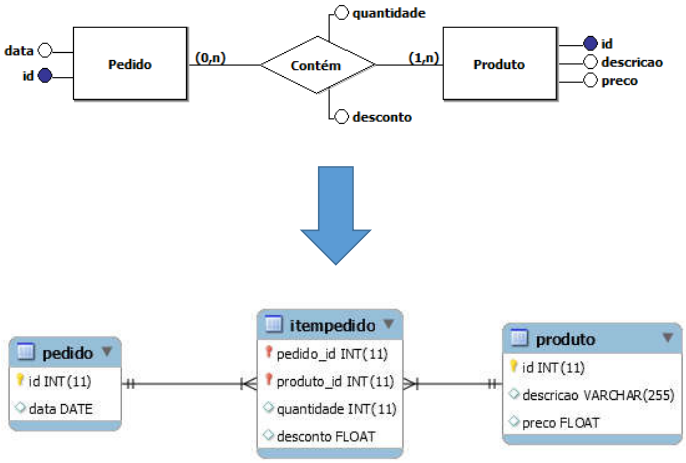
```
public class Produto {
	private Integer id;
	private String descricao;
	private Double preco;

	public Produto(Integer id, String descricao, Double preco) {
		this.id = id;
		this.descricao = descricao;
		this.preco = preco;
	}

	(...)
```

Polêmica sobre análise & design

- **Análise**: descrever o PROBLEMA (independente de paradigma e tecnologia)
- **Design**: descrever a SOLUÇÃO (preso ao paradigma)

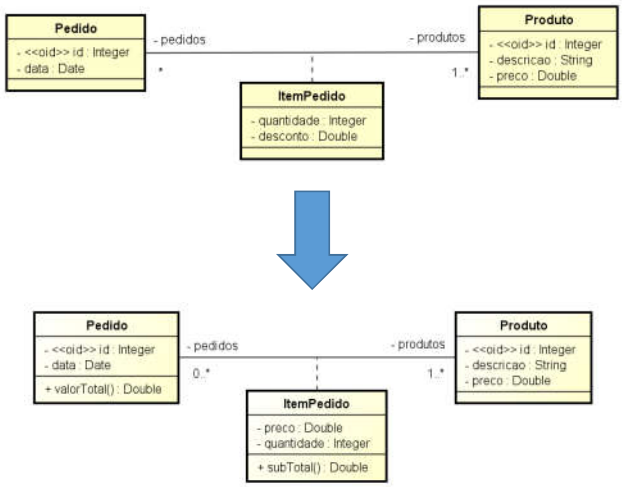


Paradigma estruturado / relacional



- Chaves estrangeiras
- Criação de tabelas
- Normalização
- Outros...

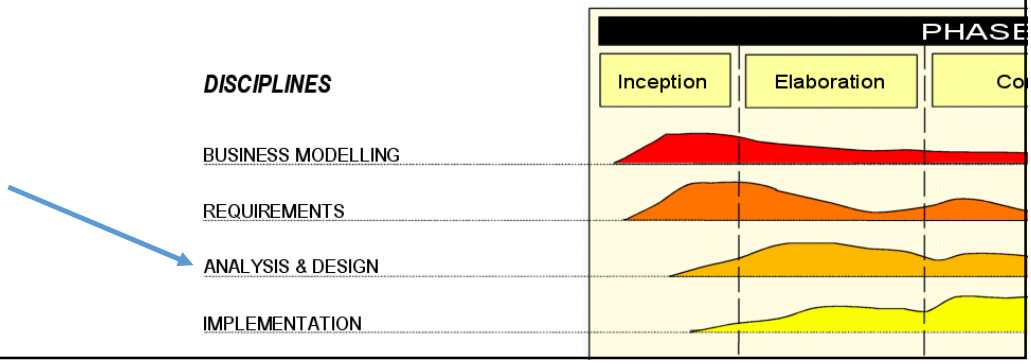
Paradigma orientado a objetos



- Inclusão de métodos
- Normalização (?)

Conclusão (polêmica)

- Análise e Design tendem a ser mais próximas no desenvolvimento orientado a objetos
- Consideraremos nossa Modelagem Conceitual como nível de Análise, mas vamos também "invadir" alguns aspectos de Design (tipos de dados, direção de associações, preocupações com normalização, etc.)



Resumo da aula

Nível	Responsável	Objetivo
Conceitual ou de Análise (de negócio)	Analista de negócio	Descrever as entidades do domínio **(do negócio)** e suas inter-relações: **Independentemente de SISTEMA**
Conceitual ou de Análise (de sistema)	Analista de sistemas	Descrever as entidades do domínio (**do sistema**) e suas inter-relações: • **Independentemente de PARADIGMA E TECNOLOGIA**
Lógico ou de Design	Projetista	Descrever as entidades do domínio (**do sistema**) e suas inter-relações: • **Preso a um PARADIGMA (ex: relacional, orientado a objetos)** • **Independentemente de TECNOLOGIA**
Físico ou de implementação	Implementador	Descrever as entidades do domínio (**do sistema**) e suas inter-relações: • **Preso a um PARADIGMA (ex: relacional, orientado a objetos)** • **Preso a uma TECNOLOGIA (ex: Java, C#, PHP, Python, Ruby, NodeJS)**

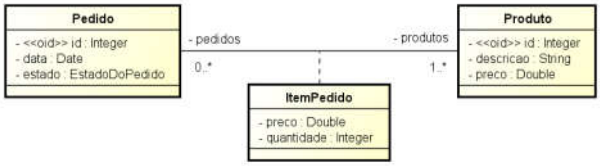
Modelo Conceitual
Conceitos e atributos

Curso: Modelagem Conceitual com Diagrama de Classes da UML
https://www.udemy.com/user/nelio-alves
Prof. Dr. Nelio Alves

Agenda

- Definição de modelo conceitual
- Conceitos
- Atributos
- Representação UML de conceitos e atributos

Modelo Conceitual

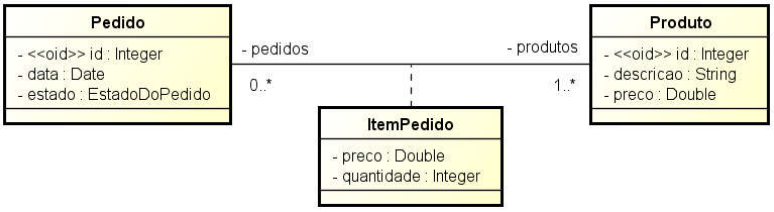


- **Definição 1:** é um modelo que descreve a estrutura das informações que o sistema vai gerenciar (Wazlawick)
- **Definição 2:** é o Modelo de Domínio em nível de Análise:
  - Pertence ao escopo do problema e não ao escopo da solução
  - Independente de paradigma
  - Independente de tecnologia
- Para representar o Modelo Conceitual, vamos utilizar a ferramenta:
  - Diagrama de Classes da UML

Modelo de domínio: *modelo que descreve as entidades do domínio, bem como as inter-relações entre elas.*

O Modelo Conceitual descreve:

- Conceitos
- Atributos
- Associações



Conceitos

- Um conceito pode ser qualquer entidade que tenha um **significado** para o sistema e que tenha uma necessidade de **armazenamento de dados**.
  - Exemplos: cliente, pedido, produto, fornecedor, etc.
- Um conceito deve ser uma **unidade coesa**.

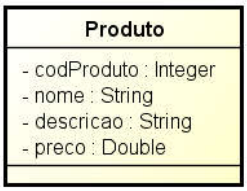
Não se deve misturar informações de várias coisas distintas em um mesmo conceito

 ?

Atributos

- Informações alfanuméricas simples, como números, textos, datas, etc. contidas em cada conceito.
  - Produto: descrição, preço
  - Cliente: nome, email, telefone, CPF, dataNascimento
- Notas (1FN):
  - Não pode ser multivalorado
    - RUIM: telefones *("3736-3938, 9988-3346, 3210-3939")*
  - Não pode ser composto
    - RUIM: endereço *("Rua Floriano Peixoto, n° 250, apto 302, Bairro Copacabana, CEP 38410-384")*
    - BOM: logradouro, numero, complemento, bairro, cep

Usando diagrama de classes da UML para representar conceitos e atributos

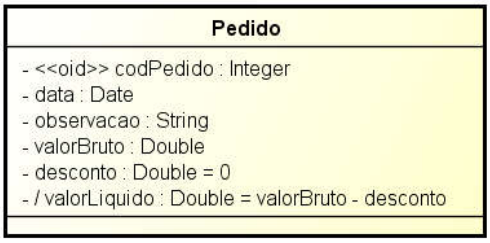


*Embora usamos os tipos da linguagem Java (por causa da ferramenta CASE usada), modelagem conceitual é uma modelagem **essencial***

REGRAS BÁSICAS:

- Um conceito é representado por um retângulo dividido em três seções
- A primeira seção contém o nome do conceito
- A segunda seção contém os atributos
- Cada atributo é representado por:
  - nome : tipo
- O tipo é opcional no MC
- A terceira seção não é usada no MC

Usando diagrama de classes da UML para representar conceitos e atributos

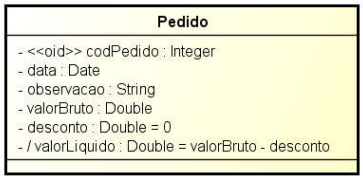


OUTRAS REGRAS:

- Atributo identificador: estereótipo <<oid>>
- Pode haver valor inicial
- Pode haver atributos derivados (read only)

Resumo da aula

- O que é modelo conceitual
  - Modelo que descreve a estrutura das informações gerenciadas pelo sistema
  - Modelo de domínio em nível de análise
  - Pertence ao escopo do problema
- Conceitos
  - algo que tenha significado para o negócio e necessidade de armazenamento
  - unidade coesa do negócio
- Atributos
  - informações alfanuméricas simples
  - não pode ser multivalorado (1FN)
  - não pode ser composto (1FN)



- Representação de conceitos e atributos com diagrama de classes da UML
  - nome : tipo
  - Atributo identificador
  - Valor inicial
  - Atributo derivado

Como identificar conceitos

Curso: Modelagem Conceitual com Diagrama de Classes da UML
https://www.udemy.com/user/nelio-alves
Prof. Dr. Nelio Alves

Agenda

- Onde buscar informações
- Orientações para identificação de conceitos
- Exemplos

Onde buscar informações

Analista de sistemas

negócio + requisitos do sistema

DEVE-SE BUSCAR INFORMAÇÕES EM:

- Documentos produzidos pela análise de requisitos do sistema:
  - Visão geral do sistema
  - Casos de uso

 ?

- Processos de negócio
- Regulamentos / normas / leis
- Documentos de registro
- Papéis e atribuições
- Práticas e ritos estabelecidos
- Conhecimento tácito (entrevistas)
- Outros

Visão geral do sistema: documento de formato livre que especifica, em linhas gerais, os requisitos do sistema.

Sistema de Controle Escolar

Deseja-se construir um sistema acadêmico. Para isso, são registrados os cursos disponíveis, onde cada um possui um nome, carga horária e valor. Quando um curso vai ser oferecido, é registrada uma turma, informando os seguintes dados: número da turma, data de início e número de vagas. Uma matrícula de um aluno em uma turma consiste na data de matrícula e no número de prestações em que o aluno vai pagar o curso. Para cada aluno, é necessário cadastrar seu nome, cpf, e data de nascimento.

Cada aluno passa por várias avaliações durante o desenrolar do curso que está cursando. Uma avaliação possui nota e data. Depois que a avaliação ocorre, é registrado resultado de cada aluno da turma (a nota que ele tirou). Um aluno é aprovado em um curso se sua nota total for pelo menos 70% da nota prevista do curso.

É importante saber a porcentagem de aprovação por turma e por curso (considerando somente as turmas que já finalizadas). Deseja-se saber também a nota final de um aluno em um curso que ele cursou, e se ele foi aprovado ou não no curso. Além disso, o sistema deve ser capaz de saber os alunos aprovados e reprovados em uma turma, bem como o aluno com melhor desempenho da turma (pode haver empates).

Caso de uso: documento estruturado que especifica uma funcionalidade do sistema por meio da **troca de informações** entre usuários (atores de sistema) e o sistema.

Comprar Livros

Cenário principal
*1. [IN] O **comprador** informa sua identificação.*
*2. [OUT] O **sistema** informa os livros disponíveis para venda (título, capa e preço) e o conteúdo atual do carrinho de compras.*
*3. [IN] O **comprador** seleciona os livros que deseja comprar.*
*4. O **comprador** decide se finaliza a compra ou se guarda o carrinho:*
4.1 Variante: Finalizar a compra.
4.2 Variante: Guardar carrinho.

Variante 4.1: Finalizar a compra
*4.1.1. [OUT] O **sistema** informa o valor total dos livros e apresenta as opções de endereço cadastradas.*
*4.1.2. [IN] O **comprador** seleciona um endereço para entrega.*
*4.1.3. [OUT] O **sistema** informa o valor de frete e total geral, bem como a lista de cartões de crédito já cadastrados para pagamento.*
*4.1.4. [IN] O **comprador** seleciona um cartão de crédito.*
*4.1.5. [OUT] O **sistema** envia os dados do cartão e valor da venda para a operadora.*
*4.1.6. [IN] A **operadora** informa o código de autorização.*
*4.1.7. [OUT] O **sistema** informa o prazo de entrega.*

Variante 4.1: Guardar carrinho
*4.2.1. [OUT] O **sistema** informa o prazo (dias) em que o carrinho será mantido.*

Exceção 1a: Comprador não cadastrado
*1a.1 [IN] O **comprador** informa seu CPF, nome, endereço e telefone.*

Exceção 4.1.2a: Endereço consta como inválido
*4.1.2a.1 [IN] O **comprador** atualiza o endereço.*
Vai para 4.1.2.

Exceção 4.1.6a: A operadora não autoriza a venda
*4.1.6a.1 [OUT] O **sistema** apresenta outras opções de cartão ao comprador.*
*4.1.6a.2 [IN] O **comprador** seleciona outro cartão.*
Vai para 4.1.5.

(Wazlawick, 2011)

Orientações para identificação de conceitos

Analista de sistemas

negócio + requisitos do sistema

Com base nas informações, identificar conceitos relevantes para o negócio e com necessidade de armazenamento.

- Documentos (ordem de serviço, orçamento)
- Pessoas (cliente, fornecedor)
- Estruturas organizacionais (departamento)
- Eventos (venda, reserva, atendimento)

Orientações para identificação de conceitos

Analista de sistemas

negócio + requisitos do sistema

Atenção! Procure por:

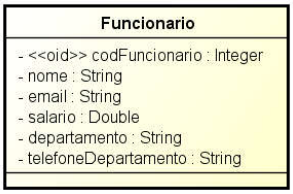
Substantivos (pessoa, compra, produto, pagamento)

Expressões que denotem substantivos (autorização de pagamento)

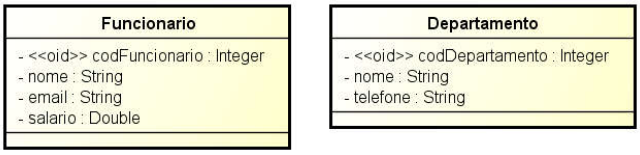
Verbos que indiquem um possível conceito (comprar, pagar)

Exemplo 1 (especificação estilo "visão geral do sistema")

Deseja-se fazer um sistema para manter um cadastro dos funcionários de uma empresa. Deseja-se poder consultar o email e salário dos funcionários, bem como o telefone de seu departamento.



ERRADO



CORRETO

Exemplo 2 ("caso de uso")

Comprar Livros

Cenário principal
*1. [IN] O **comprador** informa sua identificação.*
*2. [OUT] O **sistema** informa os livros disponíveis para venda (título, capa e preço) e o conteúdo atual do carrinho de compras.*
*3. [IN] O **comprador** seleciona os livros que deseja comprar.*
*4. O **comprador** decide se finaliza a compra ou se guarda o carrinho:*
4.1 Variante: Finalizar a compra.
4.2 Variante: Guardar carrinho.

Variante 4.1: Finalizar a compra
*4.1.1. [OUT] O **sistema** informa o valor total dos livros e apresenta as opções de endereço cadastradas.*
*4.1.2. [IN] O **comprador** seleciona um endereço para entrega.*
*4.1.3. [OUT] O **sistema** informa o valor de frete e total geral, bem como a lista de cartões de crédito já cadastrados para pagamento.*
*4.1.4. [IN] O **comprador** seleciona um cartão de crédito.*
*4.1.5. [OUT] O **sistema** envia os dados do cartão e valor da venda para a operadora.*

*4.1.6. [IN] A **operadora** informa o código de autorização.*
*4.1.7. [OUT] O **sistema** informa o prazo de entrega.*

Variante 4.1: Guardar carrinho
*4.2.1. [OUT] O **sistema** informa o prazo (dias) em que o carrinho será mantido.*

Exceção 1a: Comprador não cadastrado
*1a.1 [IN] O **comprador** informa seu CPF, nome, endereço e telefone.*

Exceção 4.1.2a: Endereço consta como inválido
*4.1.2a.1 [IN] O **comprador** atualiza o endereço.*
Vai para 4.1.2.

Exceção 4.1.6a: A operadora não autoriza a venda
*4.1.6a.1 [OUT] O **sistema** apresenta outras opções de cartão ao comprador.*
*4.1.6a.2 [IN] O **comprador** seleciona outro cartão.*
Vai para 4.1.5.

Exemplo 2 ("caso de uso")

Comprar Livros

Cenário principal
*1. [IN] O **comprador** informa sua identificação.*
*2. [OUT] O **sistema** informa os livros disponíveis para venda (título, capa e preço) e o conteúdo atual do carrinho de compras.*
*3. [IN] O **comprador** seleciona os livros que deseja comprar.*
*4. O **comprador** decide se finaliza a compra ou se guarda o carrinho:*
4.1 Variante: Finalizar a compra.
4.2 Variante: Guardar carrinho.

Variante 4.1: Finalizar a compra
*4.1.1. [OUT] O **sistema** informa o valor total dos livros e apresenta as opções de endereço cadastradas.*
*4.1.2. [IN] O **comprador** seleciona um endereço para entrega.*
*4.1.3. [OUT] O **sistema** informa o valor de frete e total geral, bem como a lista de cartões de crédito já cadastrados para pagamento.*
*4.1.4. [IN] O **comprador** seleciona um cartão de crédito.*
*4.1.5. [OUT] O **sistema** envia os dados do cartão e valor da venda para a operadora.*

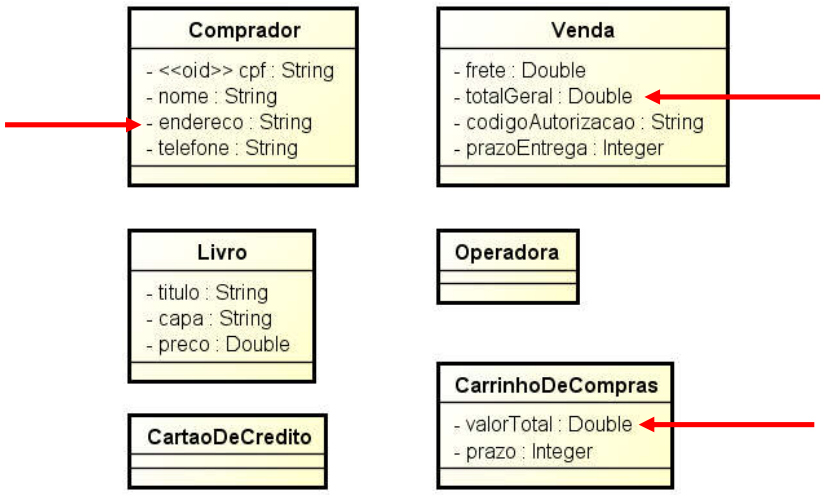
*4.1.6. [IN] A **operadora** informa o código de autorização.*
*4.1.7. [OUT] O **sistema** informa o prazo de entrega.*

Variante 4.1: Guardar carrinho
*4.2.1. [OUT] O **sistema** informa o prazo (dias) em que o carrinho será mantido.*

Exceção 1a: Comprador não cadastrado
*1a.1 [IN] O **comprador** informa seu CPF, nome, endereço e telefone.*

Exceção 4.1.2a: Endereço consta como inválido
*4.1.2a.1 [IN] O **comprador** atualiza o endereço.*
Vai para 4.1.2.

Exceção 4.1.6a: A operadora não autoriza a venda
*4.1.6a.1 [OUT] O **sistema** apresenta outras opções de cartão ao comprador.*
*4.1.6a.2 [IN] O **comprador** seleciona outro cartão.*
Vai para 4.1.5.



Requer refinamentos

Somente o caso de uso é insuficiente para se descobrir todo modelo conceitual

Resumo da aula

- Onde buscar informações
  - Documentos da análise de requisitos
    - Visão geral do sistema
    - Casos de uso
  - Processos, regulamentos, entrevistas, etc.
- Orientação:
  - Procurar conceitos com necessidade de armazenamento
  - Conceitos são substantivos
- Exemplos

Curso: Modelagem Conceitual com Diagrama de Classes da UML
https://www.udemy.com/user/nelio-alves
Prof. Dr. Nelio Alves

Capítulo: Identificação de Conceitos e Atributos

Exercícios de Fixação

Exercício 1 (RESOLVIDO): Deseja-se construir um sistema para manter um registro de artistas musicais e seus álbuns. Cada álbum possui várias músicas, as quais poderão ser consultadas pelo sistema. O sistema também deve permitir a busca de artistas por nome ou nacionalidade. O sistema também deve ser capaz de exibir um relatório dos álbuns de um artista, o qual pode ser ordenado por nome, ano, ou duração total do álbum. Um álbum pode ter a participação de vários artistas, sem distinção. Já a música pode possuir um ou mais autores e intérpretes (todos considerados artistas).

Exercício 2: Deseja-se construir um sistema para gerenciar as informações de campeonatos de handebol, que ocorrem todo ano. Deseja-se saber nome, data de nascimento, gênero e altura dos jogadores de cada time, bem qual deles é o capitão de cada time. Cada partida do campeonato ocorre em um estádio, que possui nome e endereço. Cada time possui seu estádio-sede e, assim, cada partida possui um time mandante (anfitrião) e o time visitante. O sistema deve ser capaz de listar as partidas já ocorridas e não ocorridas de um campeonato. O sistema deve também ser capaz de listar a tabela do campeonato, ordenando os times por classificação, que é calculada em primeiro lugar por saldo de vitórias e em segundo lugar por saldo de gols.

Exercício 3: Deseja-se fazer um sistema de rede social. Nesta rede social, os usuários podem seguir e ser seguidos por outros usuários. O perfil do usuário deve permitir cadastrar nome, email, data de nascimento, website, gênero, telefone e foto do perfil. Os usuários podem fazer postagens de texto em sua própria "linha do tempo" (*timeline*) da rede social, sendo que podem anexar também fotos às postagens. Uma foto é referenciada pela URI de seu local de armazenamento. As fotos podem ser organizadas em álbuns, sendo que cada álbum possui um título.

Associações

Curso: Modelagem Conceitual com Diagrama de Classes da UML
https://www.udemy.com/user/nelio-alves
Prof. Dr. Nelio Alves

Agenda

- Instâncias
- O que são associações
- Exemplo inicial

Instâncias

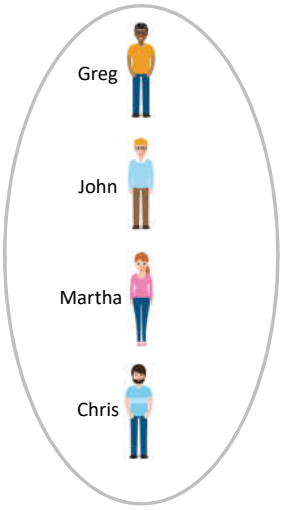
Cada ocorrência dos meus conceitos recebe o nome de INSTÂNCIA ou OBJETO

Exemplo:

Desejo criar um sistema para armazenar informações de pessoas e carros.

Conceitos:

- Pessoa
- Carro



O que são associações

Associação é um relacionamento estático entre dois conceitos.

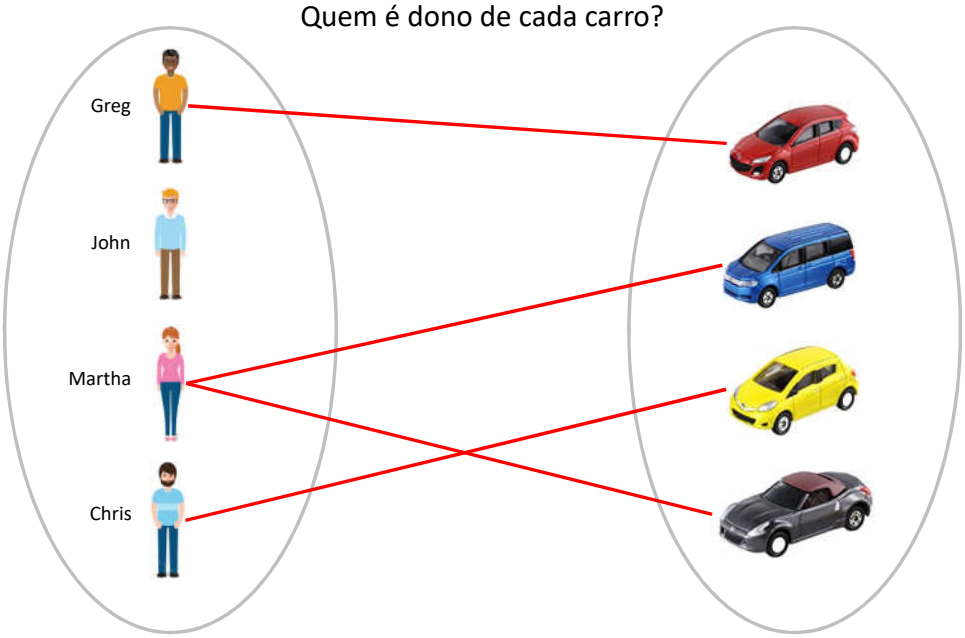
Exemplo:

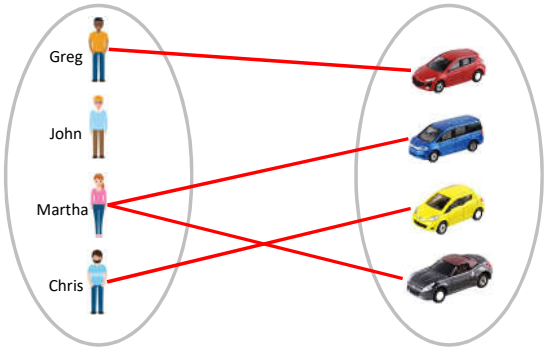
Desejo criar um sistema para armazenar informações de pessoas e carros.

Mas eu não tenho simplesmente a necessidade de saber quais são as pessoas e quais são os carros:

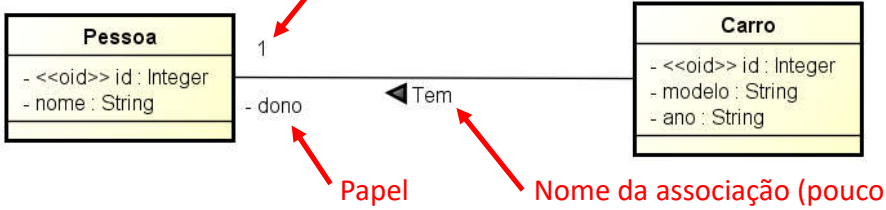
Também desejo saber quem é o dono de cada carro!

Quem é dono de cada carro?



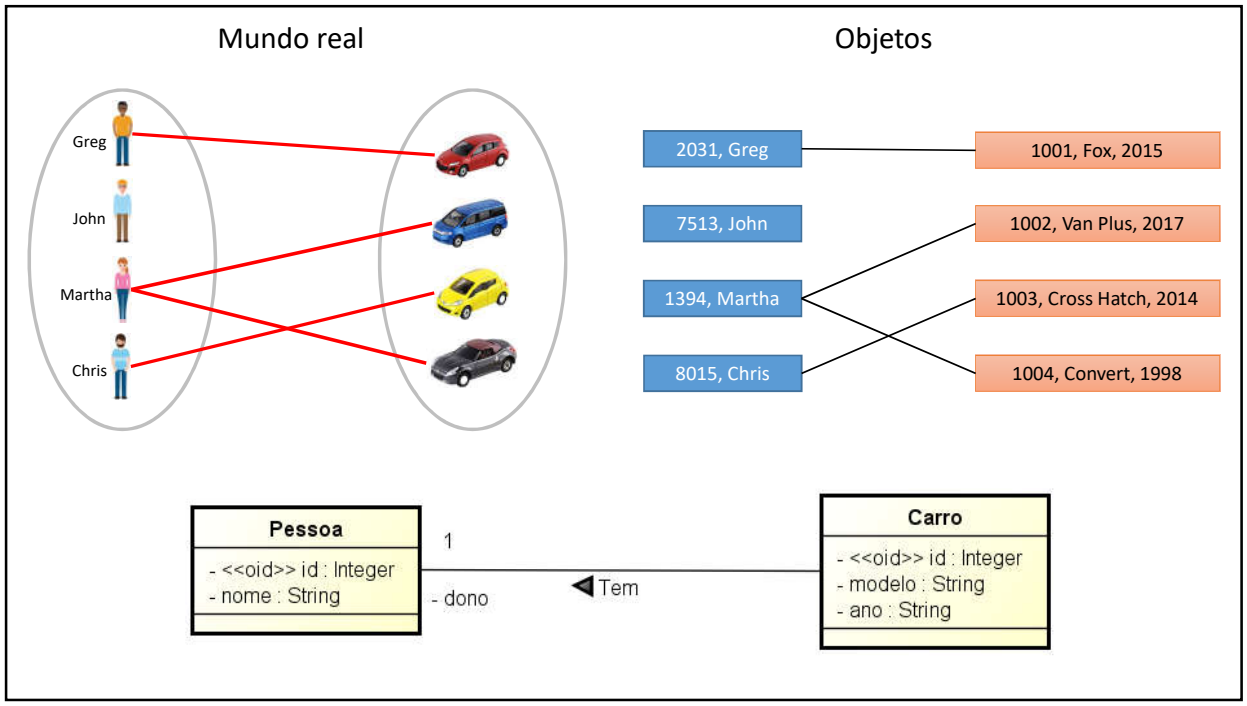


Multiplicidade (próxima aula)



Papel

Nome da associação (pouco valor)



Atenção!

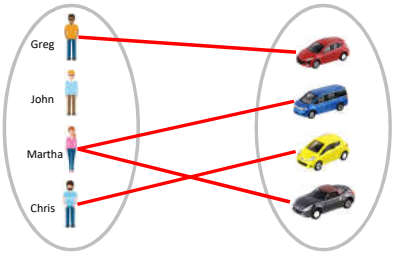
ERRADO

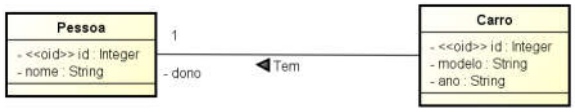
MUITO ERRADO

CORRETO

Resumo da aula

- Associação é um relacionamento estático entre dois conceitos





- Nome da associação (pouco valor)
- Nome de papel
- Multiplicidade (próxima aula)

- Não confunda com Modelo Relacional

Multiplicidade

Curso: Modelagem Conceitual com Diagrama de Classes da UML
https://www.udemy.com/user/nelio-alves
Prof. Dr. Nelio Alves

Agenda

- O que é multiplicidade?
- Como encontrar as multiplicidades?
- Associações comuns
  - Um para muitos
  - Um para um
  - Muitos para muitos

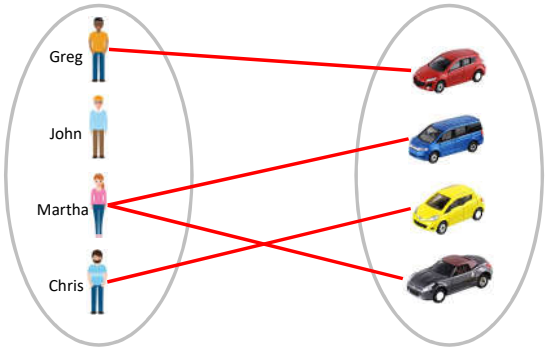
O que é multiplicidade?

É a quantidade mínima e máxima de objetos que uma associação permite em cada um de seus papéis.

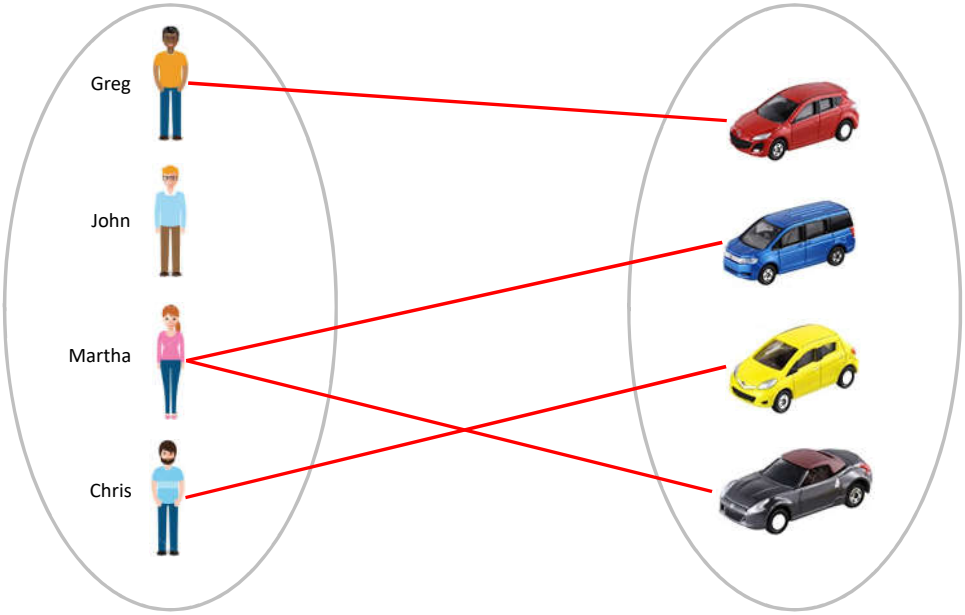
Exemplo: um carro pode ter quantos donos?

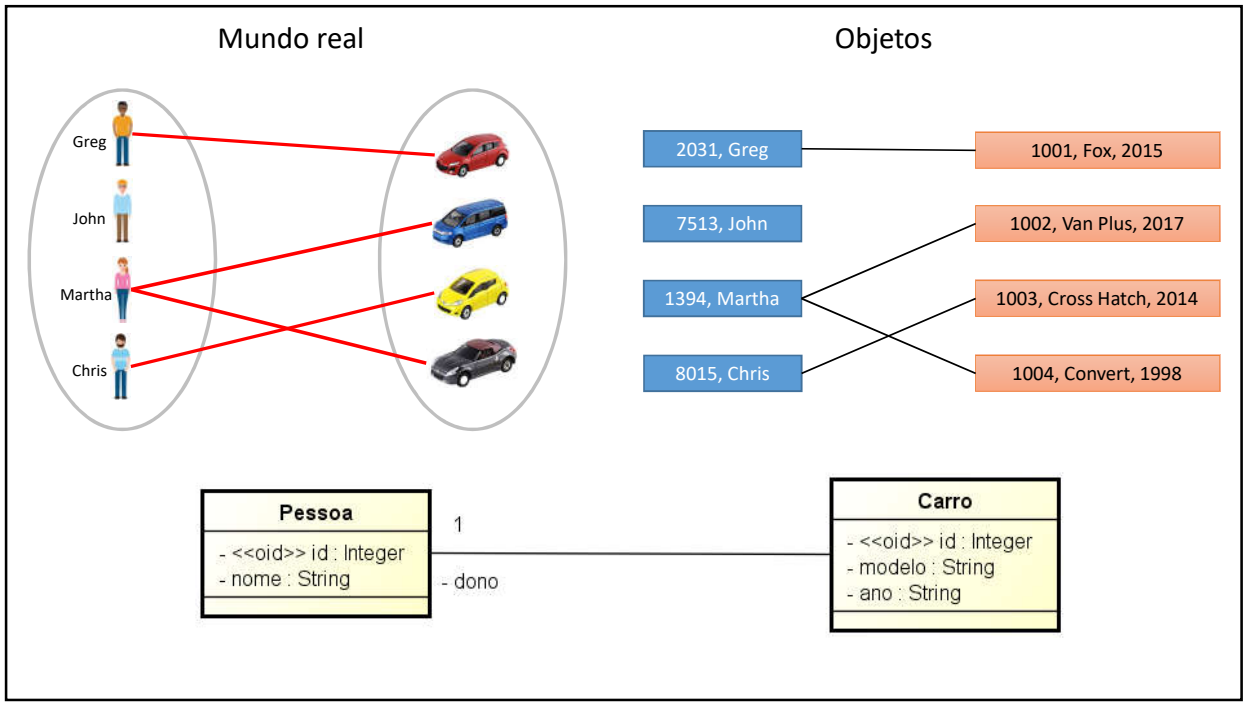
Mínimo: 1

Máximo: 1

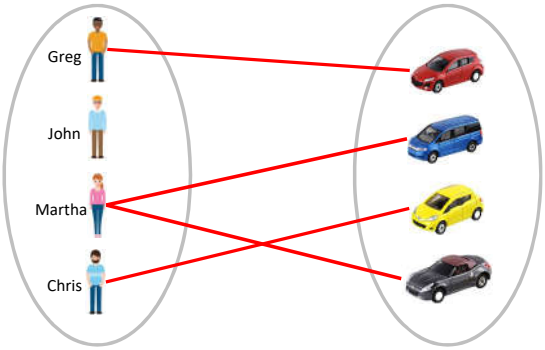


Recordando: quem é dono de qual carro?





Como encontrar as multiplicidades?



Pergunte para os dois lados:

1 <conceito> **pode ter** quantos <papel>?

1 carro **pode ter** quantos donos?

1 pessoa **pode ter** quantos carros?

Atenção: sempre inicie a pergunta com "1"

Multiplicidades possíveis

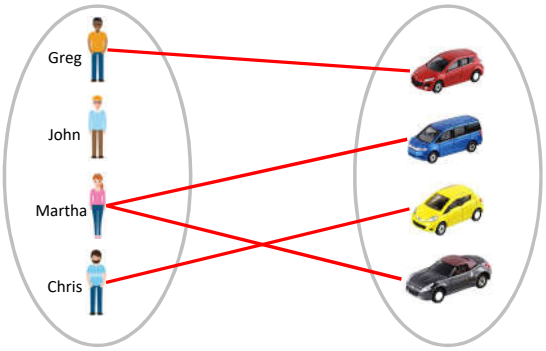
"," significa "ou"
".." significa "a"
"*" significa "vários" (sem limite específico)

a)	1	exatamente um
b)	2	exatamente dois
c)	0..1	zero a um
d)	0..*	zero ou mais
e)	*	zero ou mais
f)	1..*	um ou mais
g)	2..*	dois ou mais
h)	2..5	de dois a cinco
i)	2,5	dois ou cinco
j)	2,5..8	dois ou cinco a oito

Associações comuns

Um para muitos

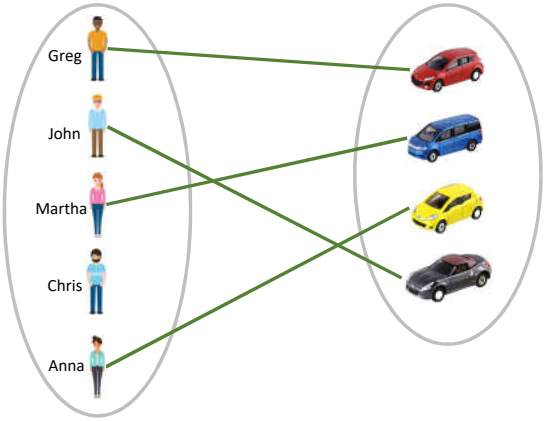
Exemplo: quem é dono de cada carro?



- Em um dos lados o máximo é 1
- No outro lado o máximo é "vários"

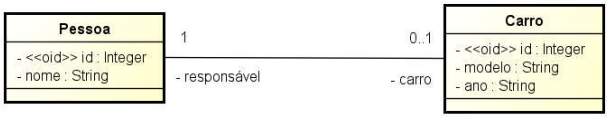
Um para um

Exemplo: quem é o responsável por cada carro?



1 carro **pode ter** quantos responsáveis?

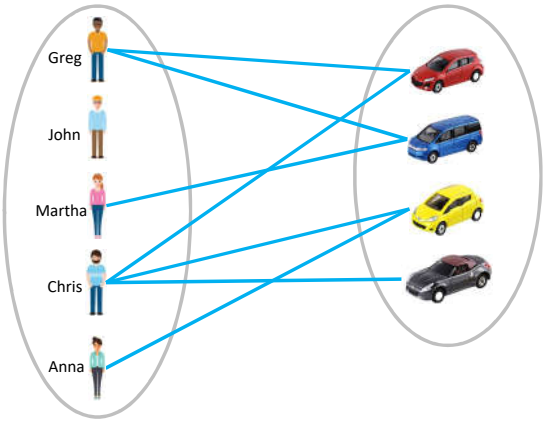
1 pessoa **pode ser responsável por** quantos carros?



- Em ambos os lados o máximo é 1

Muitos para muitos

Exemplo: quem dirige cada carro?



1 carro **pode ter** quantos motoristas?

1 pessoa **pode dirigir** quantos carros?

- Em ambos os lados o máximo é "vários"

Resumo da aula

- Multiplicidade é a quantidade **mínima** e **máxima** de objetos que uma associação permite em cada um de seus papéis.
- Como encontrar as multiplicidades?
  - Pergunte para os dois lados: **1** <conceito> pode ter quantos <papel>?
- Multiplicidades possíveis
- Associações comuns
  - Um para muitos
  - Um para um
  - Muitos para muitos

Conceito dependente Associações obrigatórias, múltiplas e autoassociações

Curso: Modelagem Conceitual com Diagrama de Classes da UML
https://www.udemy.com/user/nelio-alves
Prof. Dr. Nelio Alves

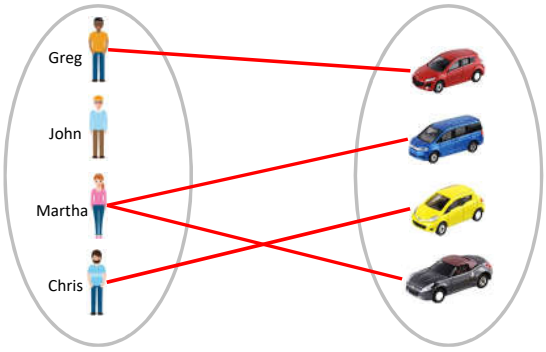
Agenda

- Associação obrigatória
- Conceito dependente
- Associações múltiplas
- Autoassociações

Associação obrigatória

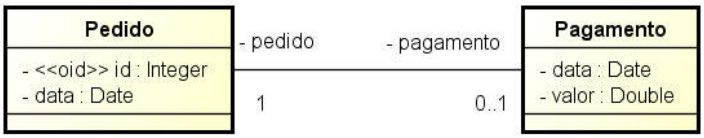
Definição

Uma associação é obrigatória se o conceito associado desempenha um papel de multiplicidade mínima maior que zero



- A associação de uma pessoa com carros não é obrigatória.
- A associação de um carro com dono é obrigatória.

Atenção



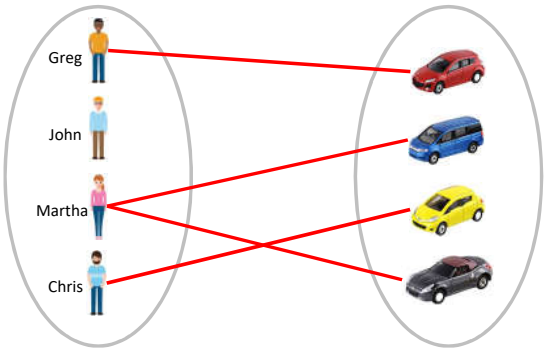
Embora o pagamento seja "obrigatório" em um pedido, um pedido pode existir temporariamente sem um pagamento.

Não confunda este caso com o conceito de associação obrigatória que acabamos de aprender.

Conceito dependente

Definição

Um conceito é dependente se ele possuir pelo menos uma associação obrigatória.



- Pessoa é um conceito independente
- Carro é um conceito dependente
  - Só pode existir se existir uma pessoa dona dele
  - Se o dono deixar de existir, o carro também deixa de existir

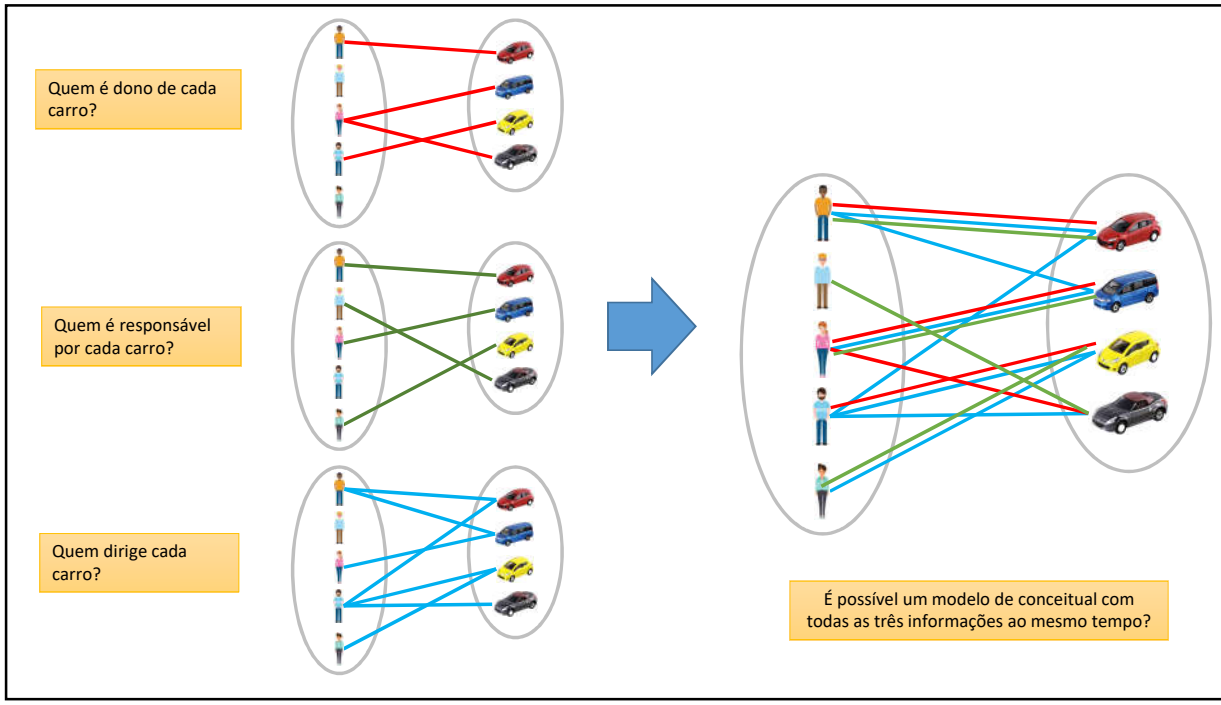
Nota

A UML tem um símbolo que denota dependência de um modo geral, mas que não acrescenta valor prático à modelagem conceitual:

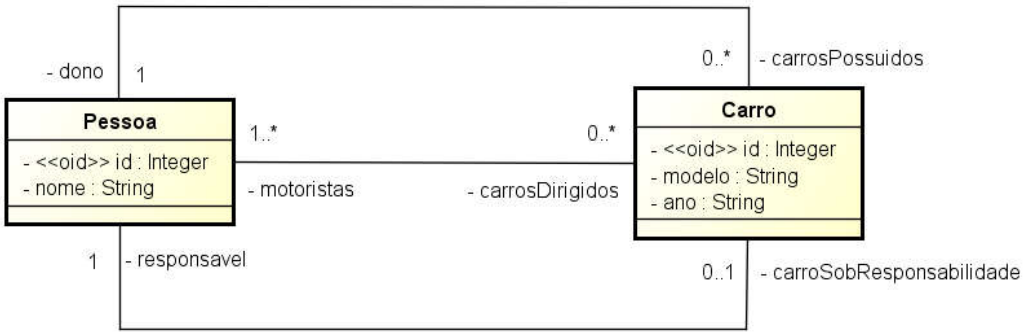
"Carro depende de pessoa"

Associações múltiplas

Pode haver mais de uma associação entre dois conceitos?



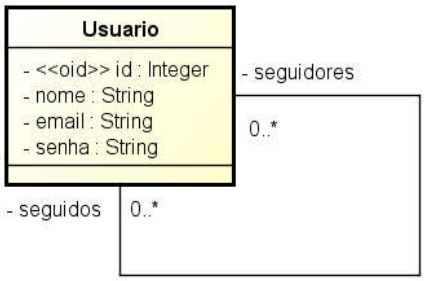
Sim. Mas os nomes de papel devem ser únicos.



Autoassociações

Autoassociações

Quando um conceito é associado com ele próprio.



1 usuário **pode ter** quantos seguidores?

1 usuário **pode ter** quantos seguidos?

Resumo da aula

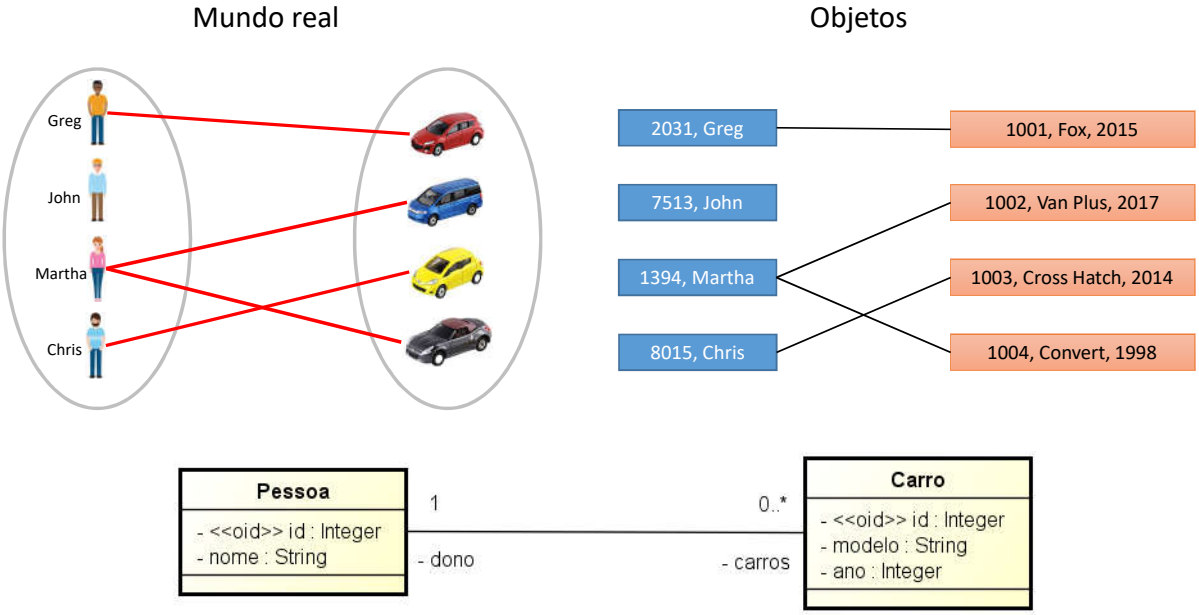
- Associação obrigatória
  - Conceito associado desempenha um papel de multiplicidade mínima maior que zero
- Conceito dependente
  - Possui pelo menos uma associação obrigatória
    - Só pode existir se o outro existir
    - Se o outro deixar de existir, o objeto dependente também deixa de existir
- Associações múltiplas
  - Ok. Nomes de papel únicos.
- Autoassociações

Desenhando instâncias com diagrama de objetos da UML

Curso: Modelagem Conceitual com Diagrama de Classes da UML
https://www.udemy.com/user/nelio-alves
Prof. Dr. Nelio Alves

Recordando

- O Modelo Conceitual representa a estrutura dos dados
  - Conceitos/atributos e como eles se inter-relacionam entre si
- Cada ocorrência de um conceito é chamada de instância ou objeto



Pra quê visualizar as instâncias (ou objetos)?

- Ajuda a compreender
- Ajuda a descobrir problemas
- Ferramenta UML: diagrama de objetos

Curso: Modelagem Conceitual com Diagrama de Classes da UML
https://www.udemy.com/user/nelio-alves
Prof. Dr. Nelio Alves

Capítulo: Associações e multiplicidade de papéis

Exercícios de Fixação

Para cada exercício, fazer:

- Desenhar o Modelo Conceitual
- Esboçar uma instância atendendo os requisitos mínimos pedidos

Exercício 1 (RESOLVIDO): Deseja-se construir um sistema para manter um registro de artistas musicais e seus álbuns. Cada álbum possui várias músicas, as quais poderão ser consultadas pelo sistema. O sistema também deve permitir a busca de artistas por nome ou nacionalidade. O sistema também deve ser capaz de exibir um relatório dos álbuns de um artista, o qual pode ser ordenado por nome, ano, ou duração total do álbum. Um álbum pode ter a participação de vários artistas, sem distinção. Já a música pode possuir um ou mais autores e intérpretes (todos considerados artistas).

Instância mínima: 2 artistas, 3 álbuns, 4 músicas

Exercício 2 (RESOLVIDO): Deseja-se construir um sistema para gerenciar as informações de campeonatos de handebol, que ocorrem todo ano. Deseja-se saber nome, data de nascimento, gênero e altura dos jogadores de cada time, bem como qual deles é o capitão de cada time. Cada partida do campeonato ocorre em um estádio, que possui nome e endereço. Cada time possui seu estádio-sede e, assim, cada partida possui um time mandante (anfitrião) e o time visitante. O sistema deve ser capaz de listar as partidas já ocorridas e não ocorridas de um campeonato. O sistema deve também ser capaz de listar a tabela do campeonato, ordenando os times por classificação, que é calculada em primeiro lugar por saldo de vitórias e em segundo lugar por saldo de gols.

Instância mínima: 1 campeonato, 2 partidas, 2 times, 2 jogadores em cada time

Exercício 3: Deseja-se fazer um sistema de rede social. Nesta rede social, os usuários podem seguir e ser seguidos por outros usuários. O perfil do usuário deve permitir cadastrar nome, email, data de nascimento, website, gênero, telefone e foto do perfil. Os usuários podem fazer postagens de texto em sua própria "linha do tempo" (*timeline*) da rede social, sendo que podem anexar também fotos às postagens. Uma foto é referenciada pela URI de seu local de armazenamento. As fotos podem ser organizadas em álbuns, sendo que cada álbum possui um título.

Instância mínima: 4 usuários, pelo menos um usuário com mais de uma postagem, pelo menos um álbum com mais de uma foto.

Exercício 4: Deseja-se construir um sistema para gerenciar as informações dos participantes das atividades de um evento acadêmico. As atividades deste evento podem ser, por exemplo, palestras, cursos, oficinas práticas, etc. Cada atividade que ocorre possui nome, descrição, preço, e pode ser dividida em vários blocos de horários (por exemplo: um curso de HTML pode ocorrer em dois blocos, sendo necessário armazenar o dia e os horários de início de fim do bloco daquele dia). Para cada participante, deseja-se cadastrar seu nome e email.

Instância mínima: 2 atividades, 4 participantes, pelo menos uma atividade com mais de um bloco de horários.

Exercício 5: Deseja-se fazer um sistema para manter dados de cidades (nome, estado, website), onde cada cidade possui um ou mais restaurantes (nome, valor da refeição) e hotéis (nome, valor da diária). Além disso, deseja-se registrar pacotes turísticos vendidos. Para registrar um pacote turístico, deve-se escolher uma cidade, definir a data da viagem, o hotel de hospedagem e o número de dias de permanência. Deve-se também definir se no pacote vai estar incluso ou não um restaurante e, se sim, quantas refeições por dia serão consumidas.

Instância mínima: 1 cidade, 2 hotéis e 2 restaurantes, 2 pacotes turísticos.

Associações todo-parte: agregação e composição

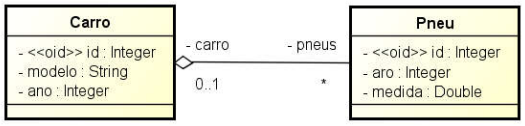
Curso: Modelagem Conceitual com Diagrama de Classes da UML
https://www.udemy.com/user/nelio-alves
Prof. Dr. Nelio Alves

Agenda

- Agregação
- Composição
- Ressalvas

Associações todo-parte

Quando um conceito é parte de outro que representa um todo, desenhamos um diamante no lado do todo.



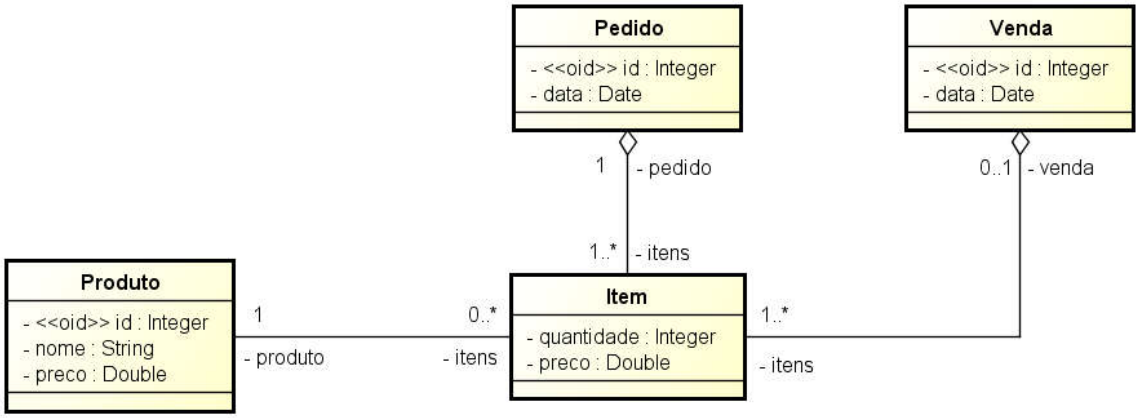
Diamante branco: "Agregação"
O conceito parte não é exclusivo

Diamante preto: "Composição"
O conceito parte é **exclusivo**

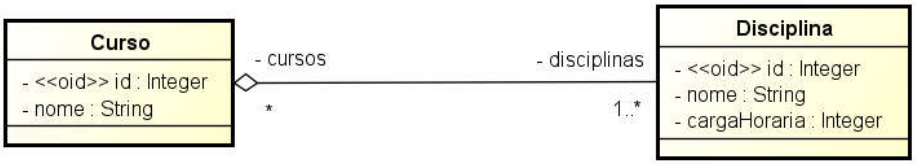
Exclusividade: 1 ou 0..1

Como a composição (diamante preto) é uma relação exclusiva, a multiplicidade no lado do diamante sempre será 1 ou 0..1

Agregação - exemplo 2

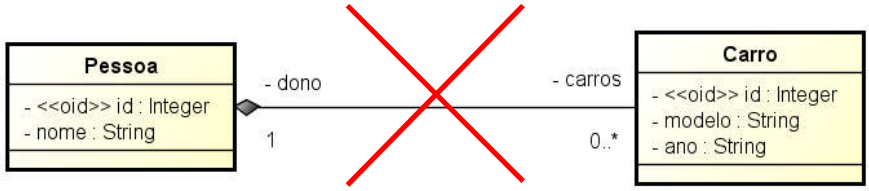


Agregação - exemplo 3



Ressalva 1

O diamante (seja branco, seja preto) deve ser usado **somente em casos em que realmente se trata de uma relação todo-parte**



ERRADO

Ressalva 2

Algumas pessoas confundem o diamante preto (composição) como se fosse uma forma de indicar a deleção em cascata dos objetos dependentes. Na verdade o que indica isso é a multiplicidade.

Resumo da aula

- Associações todo-parte
  - Agregação - diamante branco - mais fraca
  - Composição - diamante preto - mais forte - exclusiva (1 ou 0..1)
- Ressalvas
  - Use diamante somente quando realmente for uma associação todo-parte
  - Não é a composição que indica deleção em cascata

Classe de associação

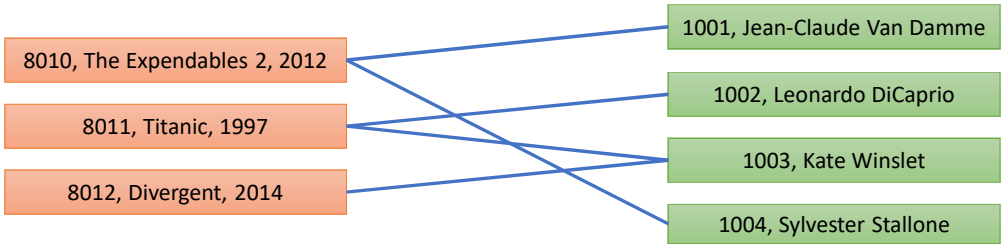
Curso: Modelagem Conceitual com Diagrama de Classes da UML
https://www.udemy.com/user/nelio-alves
Prof. Dr. Nelio Alves

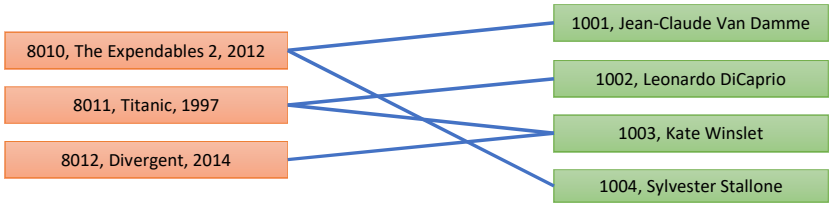
Agenda

- Exemplo motivador
- Classe de associação em associações muitos-para-muitos
- Classe de associação vs. Classe comum

Exemplo motivador

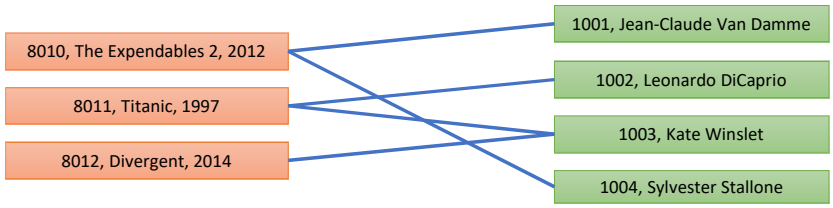
Deseja-se fazer um sistema para manter um cadastro de filmes e artistas (atores/atrizes), bem como a informação de qual artista atuou em cada filme.





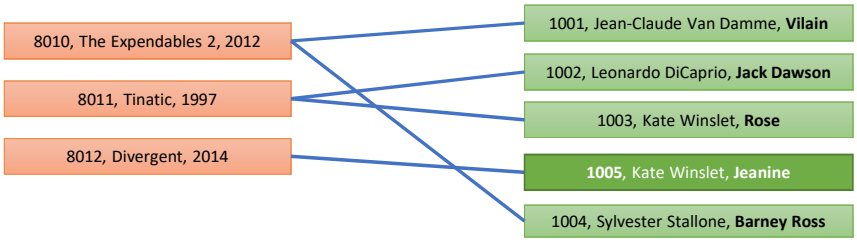
Problema

Além disso, desejo saber também o nome do personagem desempenhado por cada artista em cada filme

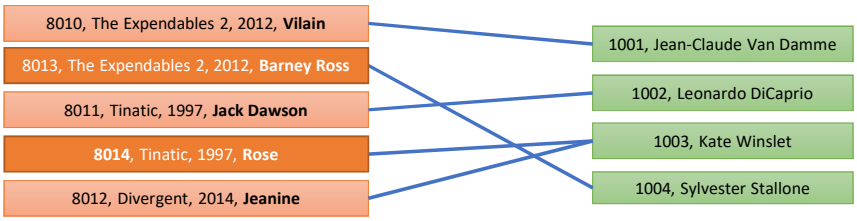


Problema

Além disso, desejo saber também o nome do personagem desempenhado por cada artista em cada filme

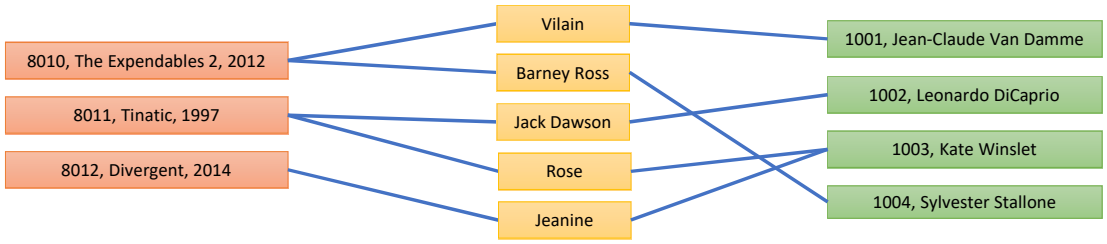


ERRADO

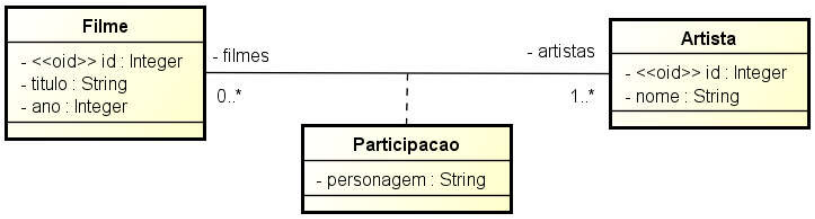


ERRADO

Conclusão: o nome do personagem é um dado que pertence à associação.
Neste caso, o nome do personagem deverá ser armazenado em um objeto intermediário.

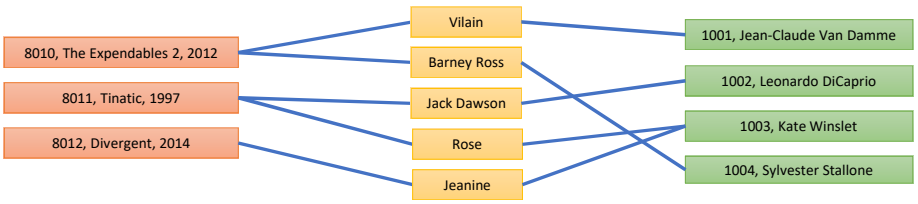


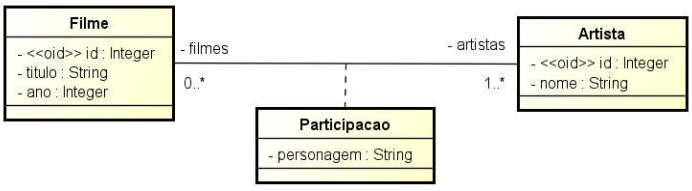
Classe de associação:



Atenção!

A classe de associação indica associação ÚNICA!

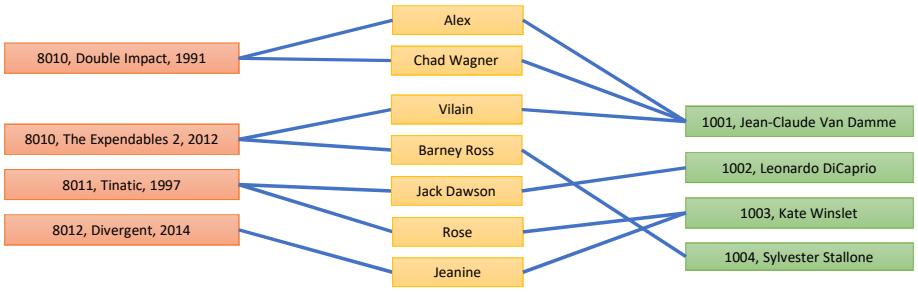


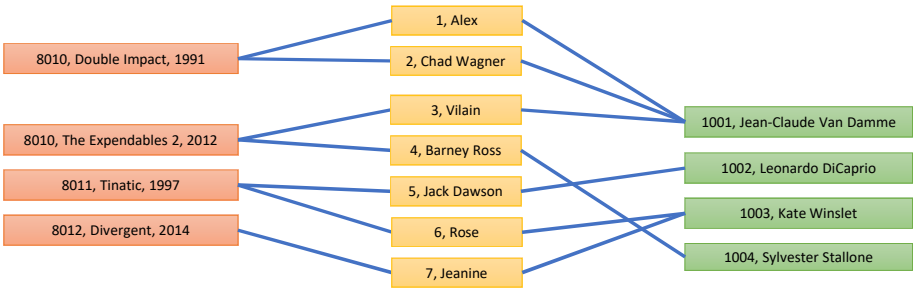


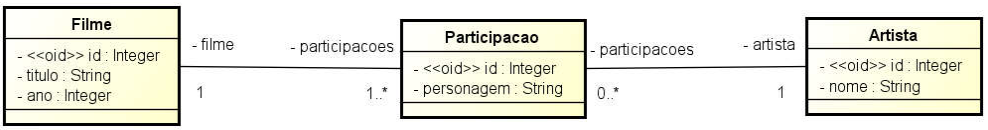
Este modelo indica que para um mesmo artista e um mesmo filme, só pode haver uma participação

Isso não pode:

Então como representar um modelo no qual um mesmo artista pode representar mais de um personagem em um mesmo filme?







Resumo da aula

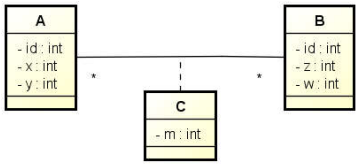
Associação muitos-para-muitos: ALERTA!
Tem atributo na associação?
SIM: então haverá um novo conceito no meio!

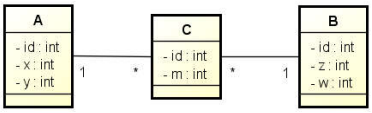
O mesmo "A" pode estar associado com o mesmo "B" mais de uma vez?

O mesmo artista pode participar do mesmo filmes mais de uma vez?

não

sim





Curso: Modelagem Conceitual com Diagrama de Classes da UML
https://www.udemy.com/user/nelio-alves
Prof. Dr. Nelio Alves

Capítulo: Associações todo-parte e classe de associação

Exercícios de Fixação

Para cada exercício, fazer:

- Desenhar o Modelo Conceitual
- Esboçar uma instância atendendo os requisitos mínimos pedidos

Exercício 1 (RESOLVIDO): Deseja-se fazer um sistema para armazenar as informações de uma locadora de jogos digitais. Cada jogo pode rodar em mais de uma plataforma (Xbox, PS3, PS4, PC, etc.). Cada jogo possui seu preço diário de locação, sendo que um mesmo jogo pode ter preços de locação diferentes para cada plataforma. Quando um cliente (nome, email, telefone, senha) deseja fazer uma locação, ele informa quais jogos ele quer locar, informando inclusive de qual plataforma é cada jogo contido na locação a ser realizada. Quando a locação é realizada, a data atual deve ser registrada para esta locação. Para cada jogo locado, o cliente informa quantos dias ele deseja ficar com cada um (note que ele pode alugar, por exemplo, um jogo X da plataforma Xbox por 2 dias e um jogo Y da plataforma PC por 5 dias, tudo para a mesma locação). A locadora também possui alguns consoles de vídeo game, os quais podem ser usados no local pelos clientes por um certo intervalo de tempo. Cada console possui um preço por cada hora (ou fração) utilizada, e contém um conjunto de acessórios (headphone, controle, Kinect, etc.).

Instância mínima: 2 plataformas, 2 jogos para cada plataforma, 2 clientes, 2 locações, 2 itens para cada locação, 2 consoles, pelo menos um console com mais de um acessório, pelo menos um cliente com mais de uma utilização de console.

Exercício 2: Deseja-se construir um sistema acadêmico. Para isso, são registrados os cursos disponíveis, onde cada um possui um nome, carga horária e valor. Quando um curso vai ser oferecido, é registrada uma turma, informando os seguintes dados: número da turma, data de início e número de vagas. Uma matrícula de um aluno em uma turma consiste na data de matrícula e no número de prestações em que o aluno vai pagar o curso. Para cada aluno, é necessário cadastrar seu nome, cpf, e data de nascimento. Cada aluno passa por várias avaliações durante o desenrolar do curso que está cursando. Uma avaliação possui nota e data. Depois que a avaliação ocorre, é registrado resultado de cada aluno da turma (a nota que ele tirou). Um aluno é aprovado em um curso se sua nota total for maior ou igual à nota mínima de aprovação prevista para o curso.

Instância mínima: 1 curso, 1 turma, 2 matrículas e 2 avaliações com resultados.

Exercício 3: Uma biblioteca deseja fazer o registro de seus empréstimos de livros. Quando um usuário pega um livro emprestado, deve ser registrada a data de empréstimo. Por padrão, o prazo de empréstimo é de dois dias, considerando atraso se o livro for devolvido depois deste tempo. Cada livro possui um título, gênero, editora e número de páginas. Um livro pode participar de uma coleção. Cada livro também possui um valor diário de multa, caso o usuário devolva o livro com atraso em relação à data prevista de devolução.

Instância mínima: 3 livros, 1 usuário, 2 empréstimos. Pelo menos um livro participando de uma coleção.

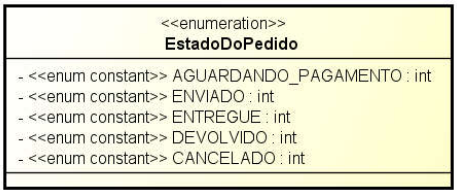
Enumerações e Tipos Primitivos

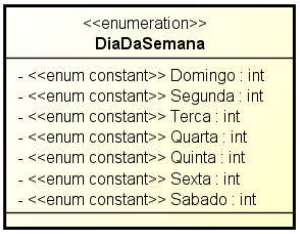
Curso: Modelagem Conceitual com Diagrama de Classes da UML
https://www.udemy.com/user/nelio-alves
Prof. Dr. Nelio Alves

Enumerações

Uma enumeração pode ser considerada um "meio termo" entre um conceito e um atributo.

Uma enumeração representa um conceito que possui um número finito de valores possíveis, valores estes que, para o negócio, valem a pena ser descritos.





Para simplificar o diagrama principal do sistema, recomenda-se incluir as enumerações em um diagrama separado, representando seu nome no diagrama do sistema diretamente como um tipo de atributo.

Diagrama de enumerações

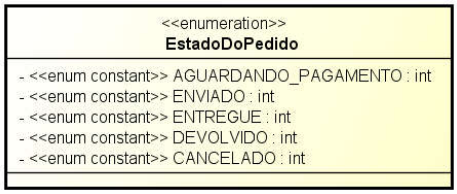
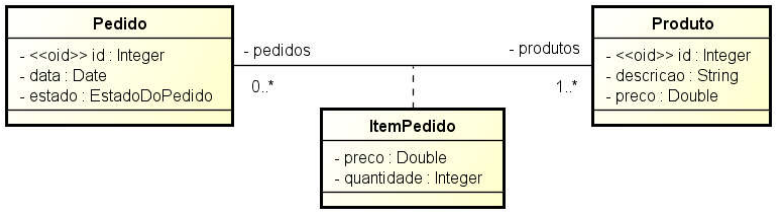


Diagrama do sistema



Tipos primitivos

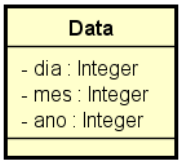
Também podem ser considerados um "meio termo" entre um conceito e um atributo.

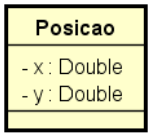
É um conceito que, devido a sua simplicidade, "não merece" ser modelado como um conceito comum no diagrama do modelo conceitual.

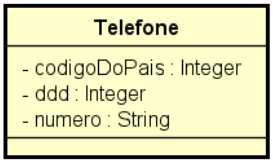
Sua regra de formação é meramente sintática e não depende de nenhum dado do sistema.

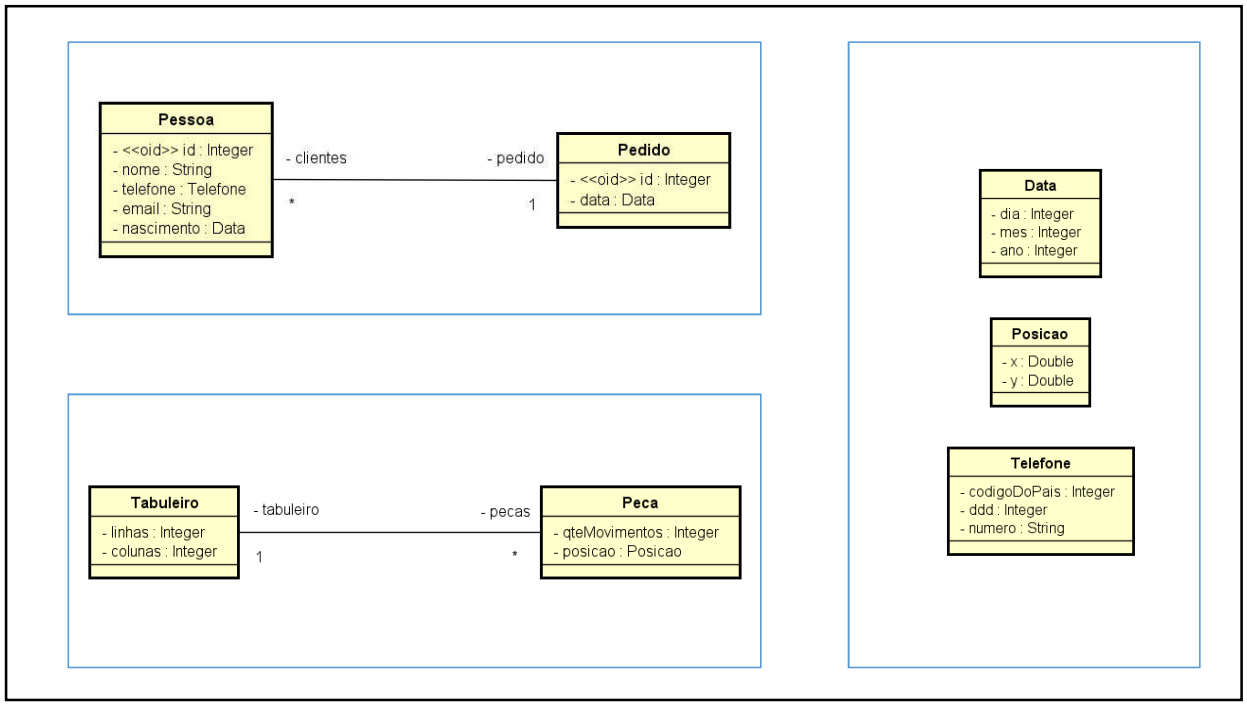
Tipicamente não possui identificador.

Outros exemplos: ISBN, CEP, Posicao3D, etc.









Ressalva

Em linguagens modernas, a data (ou data-hora) não é armazenada internamente por meio de um número inteiro para cada campo (dia, mês, ano, hora, segundo, milissegundo).

Exemplo:

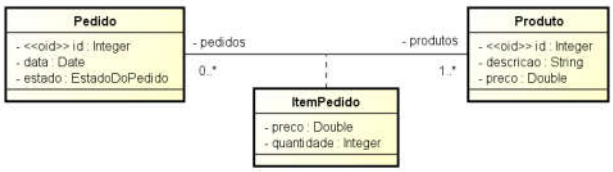
Em linguagem Java, o tipo Date, do pacote java.util, armazena uma data-hora na forma de um único número inteiro longo (long), representando a quantidade de milissegundos que se passaram desde 0:00:00 GMT 01/01/1970

Teste no compilejava.net:

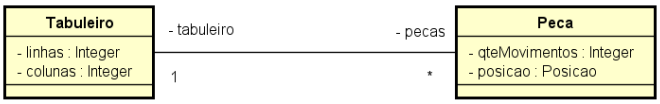
```
java.text.SimpleDateFormat sdf = new java.text.SimpleDateFormat("dd/MM/yyyy HH:mm:ss z");
java.util.Date d1 = new java.util.Date();
java.util.Date d2 = new java.util.Date(0L);
System.out.println(d1.getTime() + ": " + sdf.format(d1));
System.out.println(d2.getTime() + ": " + sdf.format(d2));
```

Resumo da aula

- Enumerações
  - Meio termo conceito/atributo
  - Conjunto finito de valores



- Tipos primitivos
  - Meio termo conceito/atributo
  - Simples
  - ISBN, CEP, Posicao, Posicao3D, etc.



Herança

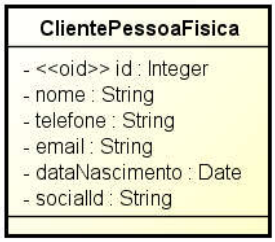
Curso: Modelagem Conceitual com Diagrama de Classes da UML
https://www.udemy.com/user/nelio-alves
Prof. Dr. Nelio Alves

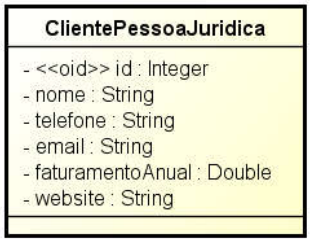
Agenda

- Exemplo motivador
- Dois primeiros questionamentos ao se considerar herança
- Definições importantes
- Ressalvas
- Quando o uso de herança é impróprio?

Exemplo motivador

Deseja-se manter um cadastro dos clientes de uma empresa, sendo que há dois tipos de clientes: pessoa física e pessoa jurídica (organizações).



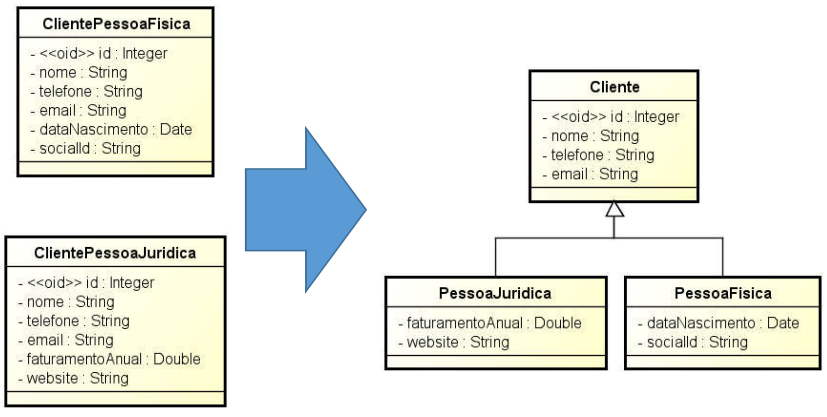


8010, Greg Junior, 3928-9211, greg@gmail.com, 12/10/1985, 83911290166

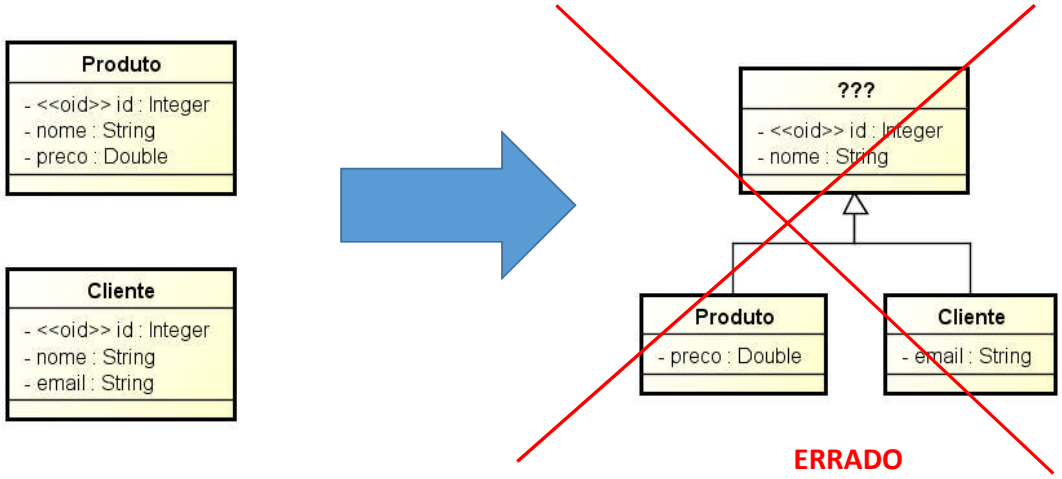
6165, Arts Inc., 30228-6160, contact@arts.com, 500000.00, www.arts.com

Questionamentos básicos ao se considerar herança:

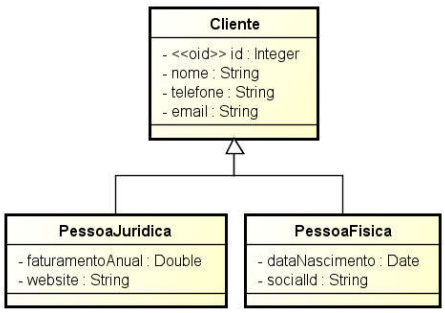
- Há estrutura comum entre os conceitos?
  - id, nome, telefone, email
- Há relação É-UM entre os conceitos e um conceito mais genérico que pode representar a estrutura comum?
  - ClientePessoaFísica é um **cliente**
  - ClientePessoaJurídica é um **cliente**



Exemplo no qual não há relação É-UM



Definições importantes



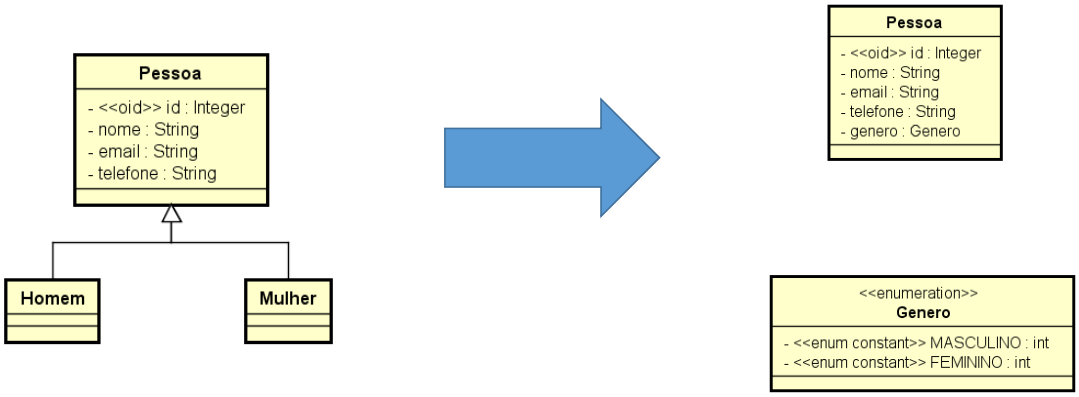
- Relação É-UM
- Relação de generalização / especialização
- Superclasse / subclasse
- Herança
  - Extensão
  - Pode adicionar elementos
  - Não pode remover elementos
- A herança é uma **associação de classes** e não de objetos (não há duas instâncias a serem ligadas)

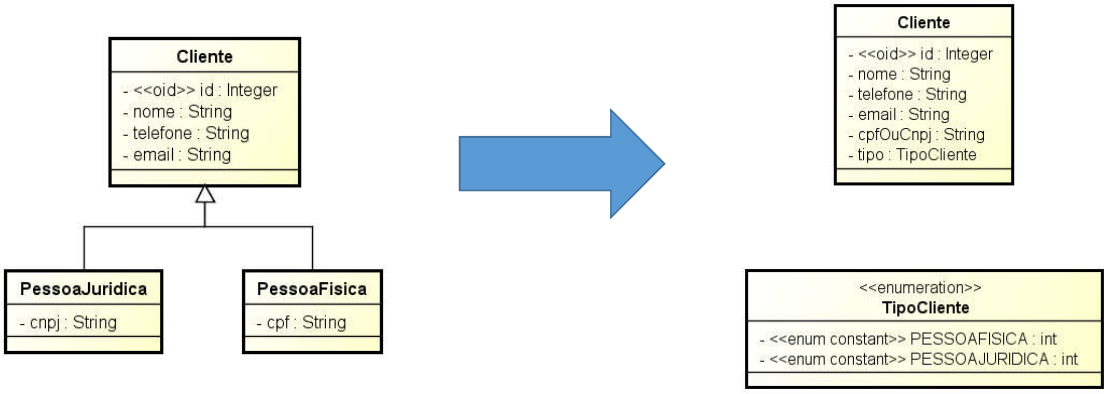
8010, Greg Junior, 3928-9211, greg@gmail.com, 12/10/1985, 83911290166

6165, Arts Inc., 30228-6160, contact@arts.com, 500000.00, www.arts.com

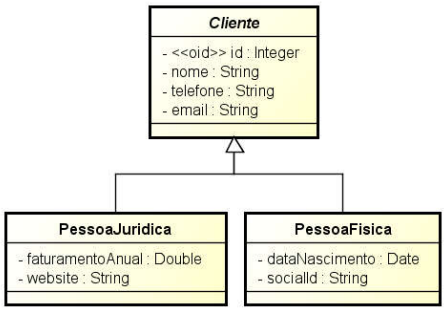
Ressalva 1

Não use herança se não há dados exclusivos de cada subclasse.





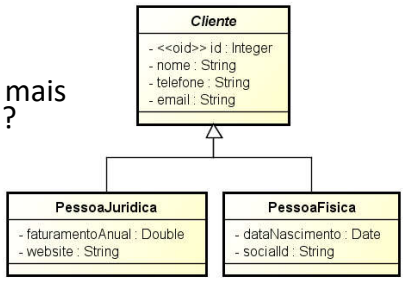
Ressalva 2



- Recomenda-se cautela com **herança parcial**, dando-se preferência a **herança total**.
- **Herança total**: somente instâncias das subclasses (PessoaJuridica e PessoaFisica) são permitidas (não são permitidas instâncias de Cliente).
- Neste caso, defina a superclasse como uma classe **abstrata** (UML = nome em itálico).

Resumo da aula

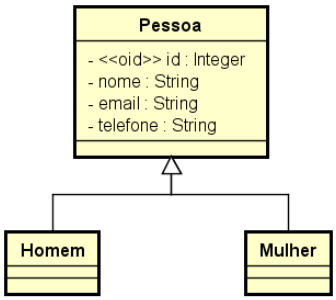
- **Questionamentos básicos ao considerar herança:**
  1. Há estrutura comum entre os dois conceitos?
  2. Há relação É-UM entre os conceitos e um conceito mais genérico que pode representar a estrutura comum?

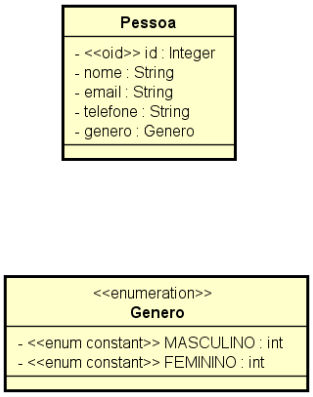


- **Definições:**
  - Relação É-UM
  - Relação generalização / especialização
  - Superclasse / subclasse
  - Herança / Extensão
  - A herança é uma associação de classes

Resumo da aula

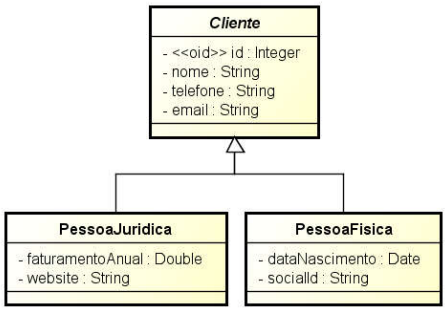
- **Ressalva 1:** Não use herança se não há dados exclusivos de cada subtipo.





Resumo da aula

- **Ressalva 2:** Prefira herança total. Faça a superclasse como **abstrata** (UML = nome em itálico)



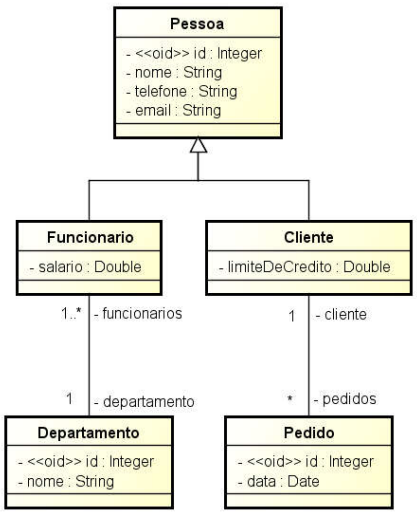
Herança - Parte 2

Curso: Modelagem Conceitual com Diagrama de Classes da UML
https://www.udemy.com/user/nelio-alves
Prof. Dr. Nelio Alves

Agenda

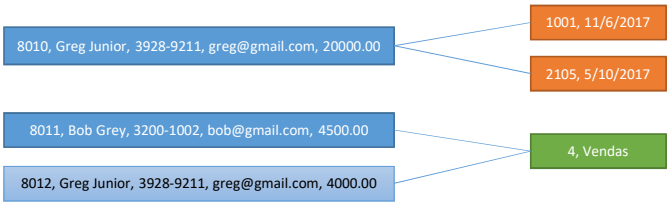
- Exemplo motivador
- Dois primeiros questionamentos ao se considerar herança
- Definições importantes
- Ressalvas
- Quando o uso de herança é impróprio?

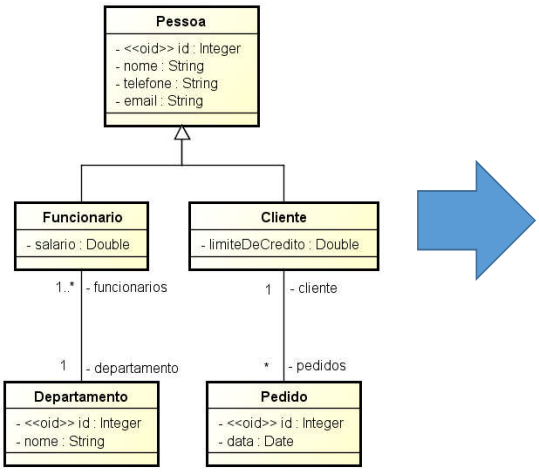
Uso impróprio de herança

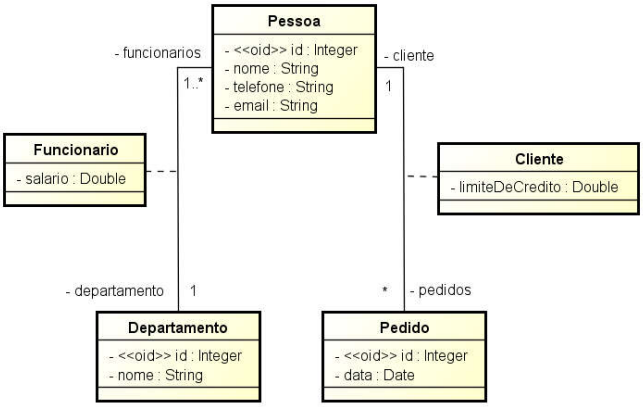


Indicativos de que o uso da herança é impróprio:

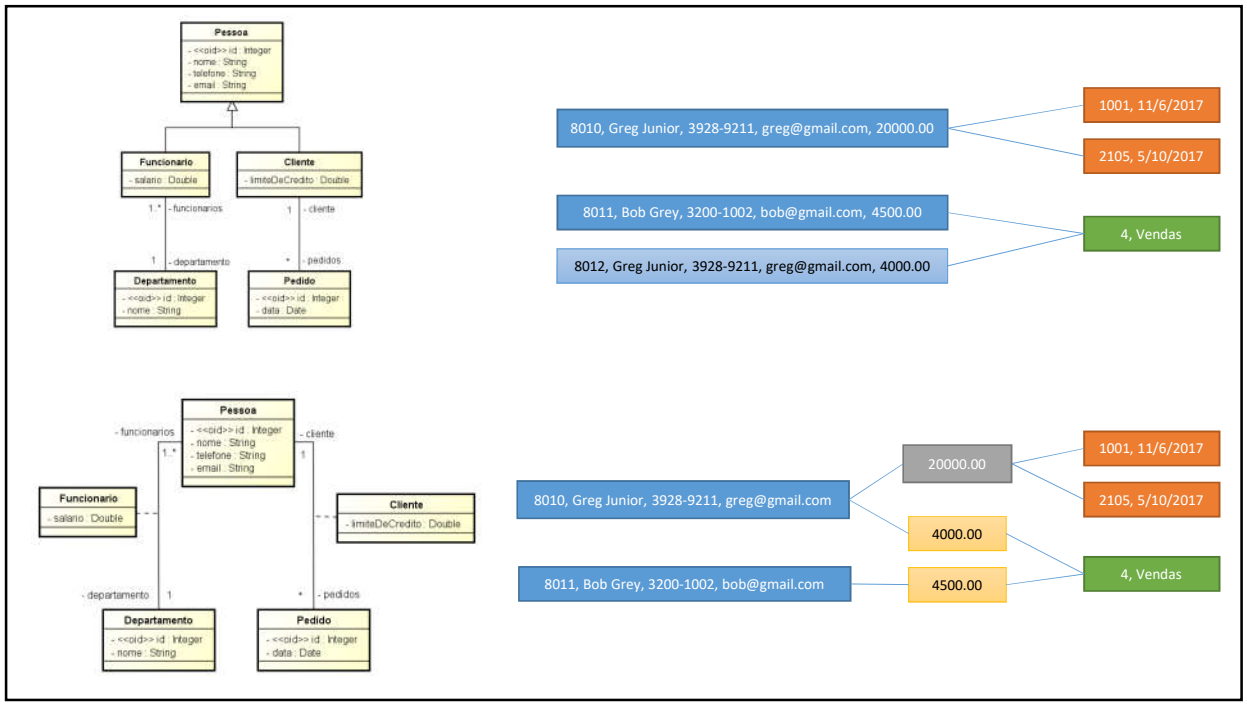
- Para o conceito da subclasse fazer sentido, ele estaria associado a outro conceito?
- Uma mesma pessoa pode ser cliente e também funcionário?







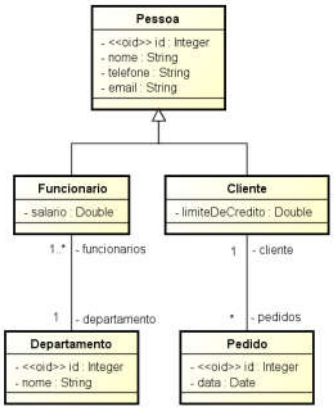
- "Funcionário" e "Cliente" não são especializações, mas sim **papéis** que uma pessoa pode assumir
- Esses papéis podem ser representados por classes de associação



Resumo da aula

- **Indicativos de que o uso da herança é impróprio:**
  1. Para o conceito da subclasse fazer sentido, ele estaria associado a outro conceito?
  2. Um mesmo <**Superclasse**> pode ser <**Conceito1**> e também <**Conceito2**>?

Impróprio



Curso: Modelagem Conceitual com Diagrama de Classes da UML
https://www.udemy.com/user/nelio-alves
Prof. Dr. Nelio Alves

Capítulo: Herança, Enumerações e Tipos Primitivos

Exercícios de Fixação

Para cada exercício, fazer:

- Desenhar o Modelo Conceitual
- Esboçar uma instância atendendo os requisitos mínimos pedidos

Exercício 1 (RESOLVIDO): Uma locadora de carros deseja fazer um sistema para armazenar as informações das locações que os clientes fazem. A locadora possui diversas sedes, sendo que cada sede possui um código e uma localidade, sendo que uma localidade é armazenada internamente por dois números de ponto flutuante (exemplo: a localidade 18°56'04.4"S 48°17'29.7"W é armazenada como -18.9345654, -48.2915811). Para cada locação é necessário registrar a data e horário em que o cliente pega o carro, bem como a sede da qual o carro foi retirado. Há dois tipos de locação: a diária e a de longo período. Quando a locação for diária, deve-se armazenar o número de dias previstos para devolução do carro. Quando a locação for de longo período, deve-se armazenar a porcentagem de desconto dada no valor da diária. Em todos os casos, a data e horário em que o carro foi devolvido devem ser armazenados. Os dados de cada carro são modelo, placa, cor, ano e data de aquisição. A locadora trabalha somente com carros de cor branca, preta, cinza e vermelha. Os dados dos clientes são nome, cpf, email e seus telefones. Cada carro pertence a uma categoria, sendo que cada categoria de carro possui um valor de diária de locação.

Instância mínima: 1 cliente, 2 carros, 3 locações.

Exercício 2: Deseja-se fazer um sistema de pedidos. Um ou mais produtos podem ser vendidos em cada pedido, sendo que a cada produto pode ser dado um desconto diferente, e também cada produto pode ser vendido em uma ou mais unidades. Cada produto possui nome e preço, e pode pertencer a várias categorias. Cada pedido é feito por um cliente, que deve ter em seu cadastro nome, telefones, email, cpf ou cnpj, e um ou mais endereços, sendo que o cliente deve especificar um endereço para entrega na hora de comprar. Para um pedido, deve ser registrado o instante em que é realizado e o endereço de entrega. Um pedido deve ser pago ou por boleto, ou por cartão de crédito. No caso de boleto, deve-se armazenar a data de vencimento e a data de pagamento. No caso de cartão de crédito, deve-se armazenar o número de parcelas. Todo pagamento possui um estado (pendente, quitado ou cancelado).

Instância mínima: 1 cliente, 3 produtos, 2 pedidos, pelo menos um pedido com mais de um produto, pelo menos um pedido pago com cartão de crédito e um com boleto.

Exercício 3: Deseja-se fazer um sistema para gerenciar informações de uma companhia aérea. Para isto, deve-se manter um cadastro das pessoas no sistema, sendo que é possível ter passageiros e pilotos. Toda pessoa possui cpf, nome e sexo, enquanto que pilotos possuem um número de licença de voo, e os passageiros possuem data de nascimento e um ou mais telefones. Quando um passageiro faz uma reserva de voo, deve ser registrado o assento deste passageiro no voo. O sistema também deve manter a informação de quem foi o piloto de cada voo, além do número, data e horário do voo.

Instância mínima: 2 pilotos, 3 passageiros, 2 voos. Pelo menos um passageiro com mais de um telefone. Pelo menos um voo com mais de um passageiro. Um dos pilotos deve ser passageiro de um voo (obviamente não pilotado por ele).

Curso: Modelagem Conceitual com Diagrama de Classes da UML

https://www.udemy.com/user/nelio-alves

Prof. Dr. Nelio Alves

Capítulo: Estudo de caso: implementação Java com Spring Boot e JPA

Notas de aula do estudo de caso

Requisitos:

Necessário:

- Computador Mac, Linux ou Windows
- Conhecimento básico de operação do sistema operacional: instalação / descompactação / terminal
- Conhecimento básico de Programação Orientada a Objetos em alguma linguagem moderna (Java, C#, Python, PHP, etc.): classes, atributos, métodos, construtores, encapsulamento, elementos estáticos.

Desejável:

- Conhecimento básico de Git

Objetivo geral:

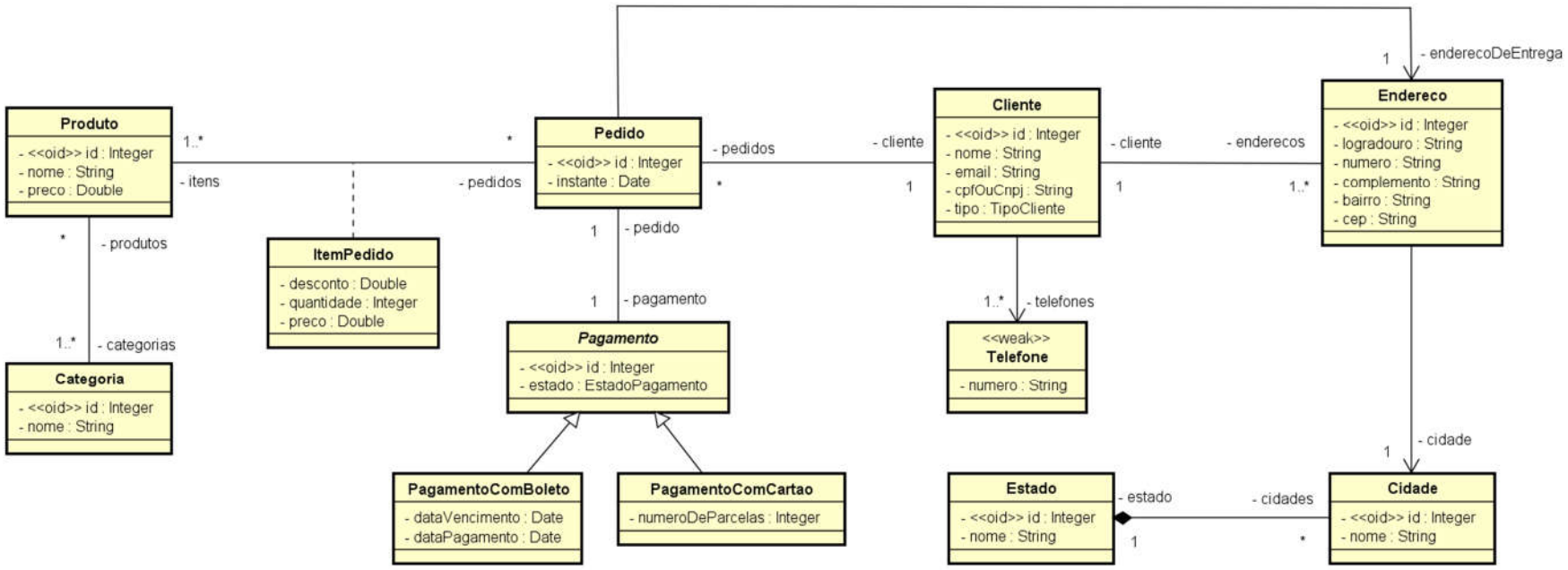
Este estudo de caso tem como objetivo mostrar na prática como um modelo conceitual pode ser implementado sobre o paradigma orientado a objetos, usando padrões de mercado e boas práticas.

Vamos tomar como caso um modelo conceitual abrangente, com o qual possamos mostrar a implementação prática em linguagem orientada a objetos dos tópicos aprendidos no curso, quais sejam:

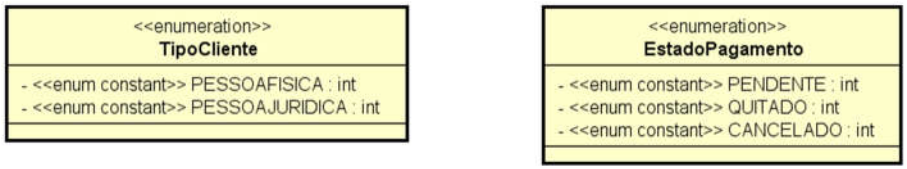
- Leitura e entendimento do diagrama de classes
- Leitura e entendimento do diagrama de objetos
- Associações
- Um para muitos / muitos para um
- Um para um
- Muitos para muitos
- Conceito dependente
- Classe de associação
- Herança
- Enumerações
- Tipos primitivos (ItemPedidoPK)
- Entidades fracas (ElementCollection)
- Associações direcionadas

Objetivos específicos:

1) Fazer uma implementação padrão do seguinte modelo conceitual:

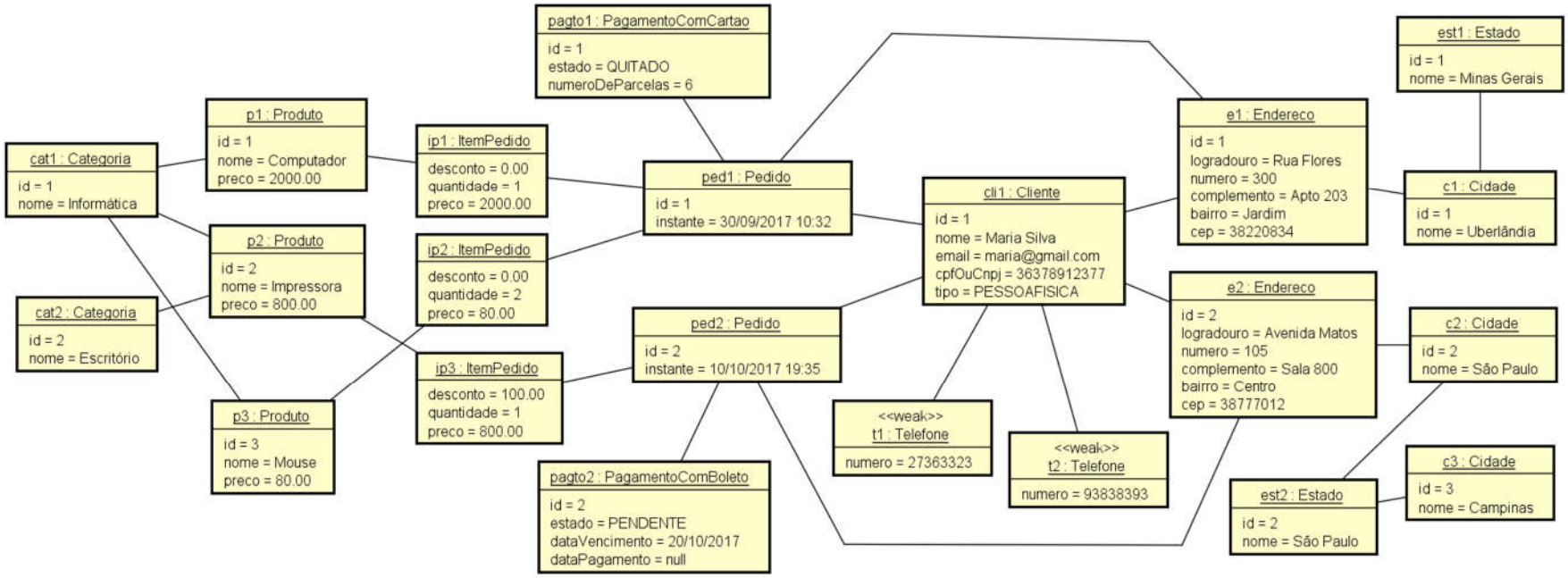


Enumerações:



Objetivos (continuação):

2) Criar a seguinte instância do modelo conceitual:



3) Gerar uma base de dados relacional automaticamente a partir do modelo conceitual, bem como povoar a base com a instância dada.

4) Recuperar os dados e disponibilizá-los por meio de uma API Rest BÁSICA. Os seguintes *end points* devem ser disponibilizados:

End point	Dados
/categorias/{id}	Categoria e seus produtos
/clientes/{id}	Cliente, seus telefones e seus endereços
/pedidos/{id}	Pedido, seu cliente, seu pagamento, seus itens de pedido, seu endereço de entrega

Instalando e testando as ferramentas

ATUALIZAÇÃO

ATENÇÃO: INSTALE A VERSÃO 8 DO JAVA JDK

- Link Java 8 JDK: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

ATENÇÃO AOS BITS DA SUA MÁQUINA (32bits ou 64bits)

- Git
- Conta no Github
- Google Chrome e Postman
- JDK - Java Development Kit
- STS - Spring Tool Suit (Eclipse / Maven / Tomcat / Jackson / JPA)

Ajuste do layout do STS:

- Window -> Perspective -> Open Perspective -> Other -> Spring
- Window -> Perspective -> Reset Perspective
- Minimizar as abas Outline e Spring Explorer

Criando e testando o projeto

ATUALIZAÇÃO

Erro comum: versão do Java JDK
Recomendação: instale o Java versão 8 e 64bits

- Vídeo: https://www.youtube.com/watch?v=HtA7EGRyPb0
- Link Java 8 JDK: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Erro comum: arquivo corrompido do Maven (invalid LOC header)
Recomendação: apague os arquivos e voltar ao STS e deixar o Maven refazer o download

- Vídeo: https://www.youtube.com/watch?v=FnI1oXbDtOg

ATENÇÃO: VERSÃO DO SPRING BOOT:
Se, na criação do projeto, você escolher a versão 2.x.x, fique atento(a) às atualizações nos inícios de algumas aulas!

As atualizações serão mostradas apenas na primeira vez em que elas forem necessárias.

- Botão direito na área da aba Package Explorer -> New -> Spring Starter Project
  - Se não aparecer: New -> Other -> Procure

- Opções:
  - Name: cursomc
  - Type: Maven
  - Java Version: 1.8
  - Group: com.nelioalves.cursomc

    - Artifact: cursomc
    - Version: 1.0.0-SNAPSHOT (é uma convenção do Maven)
    - Description: Estudo de caso Java para curso de Modelagem Conceitual com UML
    - Package: com.nelioalves.cursomc
    - Next

- Opções
    - **Spring Boot Version: 1.5.x ou 2.x.x**
    - Web -> Web

- Botão direito -> Run As -> Spring Boot App

  SE OCORRER UM ERRO PORQUE A PORTA 8080 JÁ ESTÁ EM USO, OU PARE A APLICAÇÃO, OU MUDE A PORTA:

  application.properties:

  server.port=${port:8081}

Primeiro commit: Projeto criado

- **Iniciar um repositório de versionamento na pasta do projeto:**

```
git init
```

- **Configurar usuário e email (use seu email do Github):**

```
git config --global user.name "Seu nome"
git config --global user.email "seuemail@seudominio"
```

- **Fazer o primeiro commit:**

```
git add .
git commit -m "Projeto criado"
```

Commit: Testando o REST

- **Arrumando o problema do atalho CTRL + SHIFT + O:**
    - Preferences -> General -> Keys
    - Filters -> desmarque Filter uncategorized commands
    - Localize "Go To Symbol in File", selecione-o e clique "unbind"
    - Apply / Close

- **Classe CategoriaResource (subpacote resources)**

```
package com.nelioalves.cursomc.resources;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping(value = "/categorias")
public class CategoriaResource {

	@RequestMapping(method=RequestMethod.GET)
	public String listar() {
```

```
                return "REST está funcionando!";
        }
}
```

Commit: Testando a primeira classe de dominio - Categoria

- **Checklist para criar entidades:**
  - Atributos básicos
  - Associações (inicie as coleções)
  - Construtores (não inclua coleções no construtor com parâmetros)
  - Getters e setters
  - hashCode e equals (implementação padrão: somente id)
  - Serializable (padrão: 1L)

- **Método listar atualizado:**

```
@RequestMapping(method=RequestMethod.GET)
public List<Categoria> listar() {
        Categoria cat1 = new Categoria(1, "Informática");
        Categoria cat2 = new Categoria(2, "Escritório");

        List<Categoria> lista = new ArrayList<>();
        lista.add(cat1);
        lista.add(cat2);

        return lista;
}
```

Commit: Banco de dados H2 e criação automática da base de dados

- **Dependências:**

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
</dependency>

<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
</dependency>
```

- **Rodar /h2-console com a base** jdbc:h2:mem:testdb

- **Mapeamento da classe Categoria:**

```
@Entity
public class Categoria implements Serializable {
	private static final long serialVersionUID = 1L;

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private Integer id;
```

- **Alterar o arquivo application.properties**

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.username=sa
spring.datasource.password=

spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

ATUALIZAÇÃO - H2 em algumas versões novas:

```
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console

spring.datasource.url=jdbc:h2:file:~/test
spring.datasource.username=sa
spring.datasource.password=
spring.datasource.driver-class-name=org.h2.Driver

spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true

# No JDBC URL: jdbc:h2:file:~/test
```

Commit: Criando repository e service para Categoria

ATUALIZAÇÃO

Se você criou o projeto usando Spring Boot versão 2.x.x:

Em CategoriaService, onde na aula é mostrado:

```
public Categoria find(Integer id) {
	Categoria obj = repo.findOne(id);
	return obj;
}
```

Troque pelo seguinte código (`import java.util.Optional`):

```
public Categoria find(Integer id) {
	Optional<Categoria> obj = repo.findById(id);
	return obj.orElse(null);
}
```

Documentação da classe Optional:
https://docs.oracle.com/javase/8/docs/api/java/util/Optional.html

Commit: Criando operacao de instanciacao

ATUALIZAÇÃO

Se você criou o projeto usando Spring Boot versão 2.x.x:

No programa principal, onde na aula é mostrado:

```
categoriaRepository.save(Arrays.asList(cat1, cat2));
```

Troque pelo seguinte código:

```
categoriaRepository.saveAll(Arrays.asList(cat1, cat2));
```

Commit: Produto e associacao muitos para muitos

- **Mapeamento na classe Produto:**

```
@ManyToMany
@JoinTable(name = "PRODUTO_CATEGORIA",
  joinColumns = @JoinColumn(name = "produto_id"),
  inverseJoinColumns = @JoinColumn(name = "categoria_id")
)
private List<Categoria> categorias = new ArrayList<>();
```

- **Mapeamento na classe Categoria:**

```
@ManyToMany(mappedBy="categorias")
private List<Produto> produtos = new ArrayList<>();
```

Commit: Ajustes no endpoint /categorias/{id}

ATUALIZAÇÃO

Se você criou o projeto usando Spring Boot versão 2.x.x:

Em CategoriaService, onde na aula é mostrado:

```
public Categoria find(Integer id) {
	Categoria obj = repo.findOne(id);
	if (obj == null) {
		throw new ObjectNotFoundException("Objeto não encontrado! Id: " + id
				+ ", Tipo: " + Categoria.class.getName());
	}
	return obj;
}
```

Troque pelo seguinte código:

```
public Categoria find(Integer id) {
	Optional<Categoria> obj = repo.findById(id);
	return obj.orElseThrow(() -> new ObjectNotFoundException(
			"Objeto não encontrado! Id: " + id + ", Tipo: " + Categoria.class.getName()));
}
```

- **Proteção para referência cíclica na serialização Json:**

  @JsonManagedReference
  @JsonBackReference

- **Checklist de tratamento de exceção de id inválido:**
  - Criar ObjectNotFountException
  - Criar StandardError
  - Criar ResourceExceptionHandler

Commit: Estado e Cidade

- **Checklist para criar entidades:**
  - Atributos básicos
  - Associações (inicie as coleções)
  - Construtores (não inclua coleções no construtor com parâmetros)
  - Getters e setters
  - hashCode e equals (implementação padrão: somente id)
  - Serializable (padrão: 1L)

- **Mapeamentos:**

  ```
  @Entity
  ```

```
public class Cidade  implements Serializable {
	private static final long serialVersionUID = 1L;

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private Integer id;
	private String nome;

	@ManyToOne
	@JoinColumn(name="estado_id")
	private Estado estado;


@Entity
public class Estado implements Serializable {
	private static final long serialVersionUID = 1L;

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private Integer id;
	private String nome;

	@OneToMany(mappedBy="estado")
	private List<Cidade> cidades;
```

Commit: Cliente, TipoCliente, telefones e enderecos

- **Implementação do Enum:**

```
package com.nelioalves.cursomc.domain.enums;

public enum TipoCliente {

	PESSOAFISICA(1, "Pessoa Física"),
	PESSOAJURIDICA(2, "Pessoa Jurídica");

	private int cod;
	private String descricao;

	private TipoCliente(int cod, String descricao) {
		this.cod = cod;
		this.descricao = descricao;
	}

	public int getCod() {
		return cod;
	}

	public String getDescricao() {
		return descricao;
	}

	public static TipoCliente toEnum(Integer id) {
```

```
        if (id == null) {
            return null;
        }

        for (TipoCliente x : TipoCliente.values()) {
            if (id.equals(x.getCod())) {
                return x;
            }
        }
        throw new IllegalArgumentException("Id inválido " + id);
    }

}
```

- **Definição do tipo do cliente e seu getter e setter:**

```
private Integer tipo;


public TipoCliente getTipo() {
	return TipoCliente.toEnum(tipo);
}

public void setTipo(TipoCliente tipo) {
	this.tipo = tipo.getCod();
}
```

- **Mapeamento dos telefones (ElementCollection):**

```
@ElementCollection
@CollectionTable(name ="TELEFONE")
private Set<String> telefones = new HashSet<>();
```

Commit: Endpoint /clientes/{id} disponivel

- **Checklist:**
  - Criar ClienteServico
  - Criar ClienteResource
  - Proteger contra serialização Json cíclica

Commit: Pedido, EstadoPagamento e Pagamento

- **Nota: Mapeamentos de herança:**
  https://www.thoughts-on-java.org/complete-guide-inheritance-strategies-jpa-hibernate/

- **Classe Pedido:**

```
@Entity
public class Pedido  implements Serializable {
	private static final long serialVersionUID = 1L;
```

```
	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private Integer id;

	@Temporal(TemporalType.TIMESTAMP)
	private Date instante;

	@OneToOne(cascade = CascadeType.ALL, mappedBy="pedido")
	private Pagamento pagamento;

	@ManyToOne
	@JoinColumn(name="cliente_id")
	private Cliente cliente;

	@ManyToOne
	@JoinColumn(name="endereco_id")
	private Endereco enderecoDeEntrega;
```

- **Classe Pagamento:**

```
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Pagamento implements Serializable {
	private static final long serialVersionUID = 1L;

	@Id
	private Integer id;

	private Integer estado;

	@JoinColumn(name="pedido_id")
    @OneToOne
    @MapsId
    private Pedido pedido;
```

- **Classe PagamentoComBoleto:**

```
@Entity
public class PagamentoComBoleto extends Pagamento {
	private static final long serialVersionUID = 1L;

	@Temporal(TemporalType.DATE)
	private Date dataVencimento;

	@Temporal(TemporalType.DATE)
	private Date dataPagamento;

	public PagamentoComBoleto() {
	}
```

- **Classe PagamentoComCartao:**

```
@Entity
public class PagamentoComCartao extends Pagamento {
	private static final long serialVersionUID = 1L;
```

```
        private Integer numeroDeParcelas;
```

- **Instanciação:**

```
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy hh:mm");

Pedido ped1 = new Pedido(null, sdf.parse("30/09/2017 10:32"), cli1, e1);
Pedido ped2 = new Pedido(null, sdf.parse("10/10/2017 19:35"), cli1, e2);

cli1.getPedidos().addAll(Arrays.asList(ped1, ped2));

Pagamento pagto1 = new PagamentoComCartao(null, EstadoPagamento.QUITADO, ped1, 6);
ped1.setPagamento(pagto1);

Pagamento pagto2 = new PagamentoComBoleto(null, EstadoPagamento.PENDENTE, ped2, sdf.parse("20/10/2017
00:00"), null);
ped2.setPagamento(pagto2);

pedidoRepository.save(Arrays.asList(ped1, ped2));
pagamentoRepository.save(Arrays.asList(pagto1, pagto2));
```

Commit: ItemPedido e ItemPedidoPK

- **Classe ItemPedidoPK:**

```
@Embeddable
public class ItemPedidoPK implements Serializable {
        private static final long serialVersionUID = 1L;

        @ManyToOne
        @JoinColumn(name="pedido_id")
        private Pedido pedido;

        @ManyToOne
        @JoinColumn(name="produto_id")
        private Produto produto;
```

ATENÇÃO: no hashCode e equals, incluir ambos objetos associados que identifica o item

- **Classe ItemPedido:**

```
@Entity
public class ItemPedido {

        @EmbeddedId
        private ItemPedidoPK id = new ItemPedidoPK();

        private Double desconto;
        private Integer quantidade;
        private Double preco;

        public ItemPedido() {
        }
```

```
	public ItemPedido(Pedido pedido, Produto produto, Double desconto, Integer
quantidade, Double preco) {
		super();
		id.setPedido(pedido);
		id.setProduto(produto);
		this.desconto = desconto;
		this.quantidade = quantidade;
		this.preco = preco;
	}
```

Commit: Endpoint /pedidos/{id} disponibilizado

- **Checklist:**
  - Criar PedidoServico
  - Criar PedidoResource
  - Proteger contra serialização Json cíclica

Commit: Atualizacao: utilizando somente JsonIgnore

Em teste realizados, o uso de @JsonManagedReference/@JsonBackRefence apresentou alguns problemas com o envio de dados Json em requisições .

Assim, ao invés de usar @JsonManagedReference/@JsonBackRefence, vamos simplesmente utilizar o @JsonIgnore no lado da associação que não deve ser serializada. Para isto faça:

- Para cada classe de domínio:
  - Apague as anotações @JsonManagedReference existentes
  - Troque as anotações @JsonBackRefence por @JsonIgnore