GSI511 – Programação orientada a objetos I

Profa. Alessandra Paulino

2º Período

Sumário

Relacionamentos I

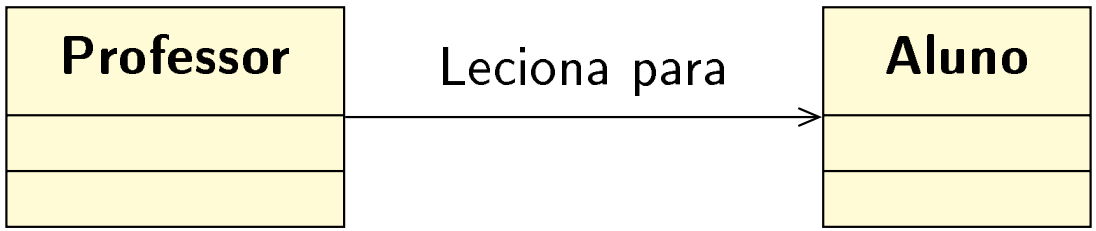
Relacionamentos são caracterizados por:

- **Nome**: descrição dada ao relacionamento (*faz*, *tem*, *possui*,...);
  - É usualmente um verbo.
- **Navegabilidade**: indicada por uma seta no fim do relacionamento;
  - Uni- ou bidirecional.
- **Multiplicidade**: 0..1, 0..*, 1, 1..*, 2, 3..7
- **Tipos de relacionamentos**: associação simples, agregação, composição, generalização.

Nomeando associações

Para facilitar seu entendimento, uma associação pode ser nomeada.

- O nome é representado como um "rótulo" colocado ao longo da linha de associação;
- Um nome de associação é usualmente um verbo ou uma frase verbal.



Associação simples I

É a forma **mais fraca** de relacionamento entre classes:

- As classes que participam desse relacionamento são independentes;
- Pode envolver duas ou mais classes.

Representa relacionamentos "usa":

- Ex.: uma pessoa "usa um" carro
- **Na implementação**: um objeto A usa outro objeto B *chamando um método público de B*.

Associação simples II

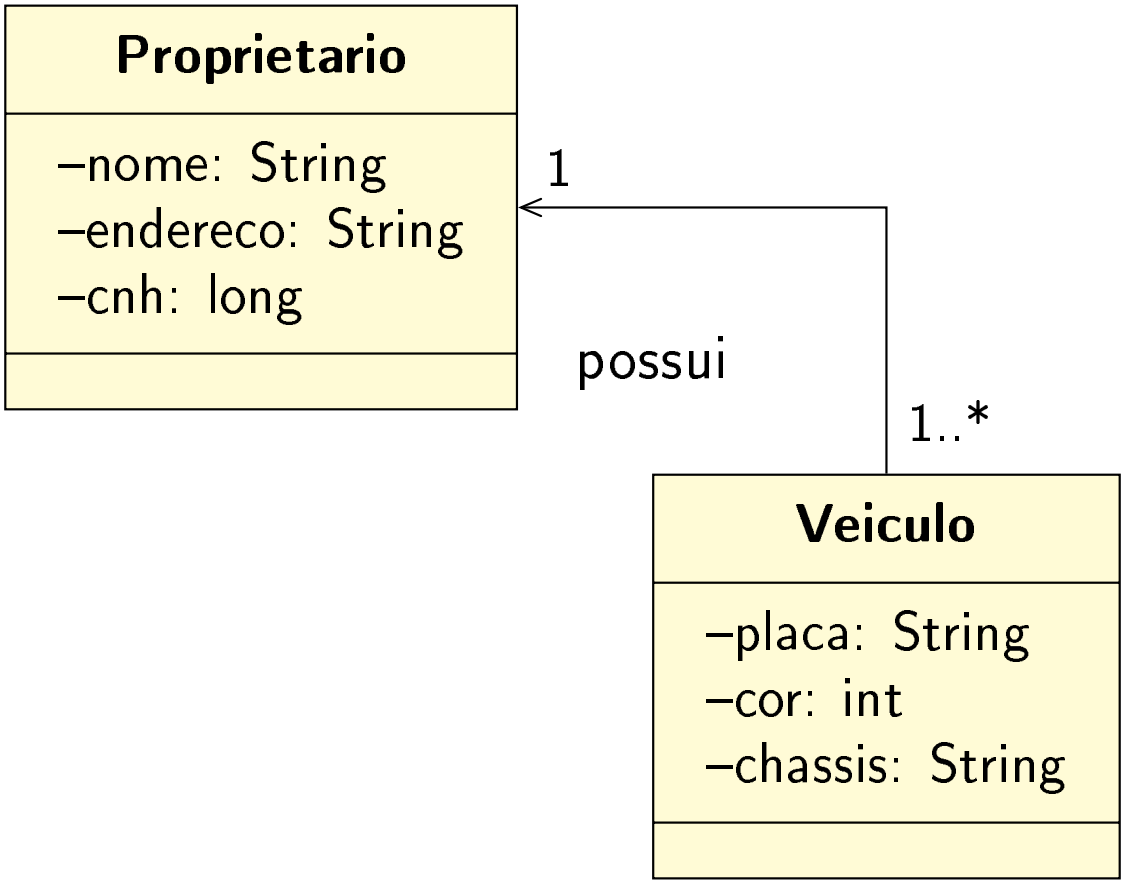
No diagrama de classes, as associações:

- São representadas como linhas conectando as classes participantes do relacionamento;
- Podem ter um nome identificando a associação;
- Podem ter uma seta junto ao nome indicando que a associação somente pode ser utilizada em uma única direção.

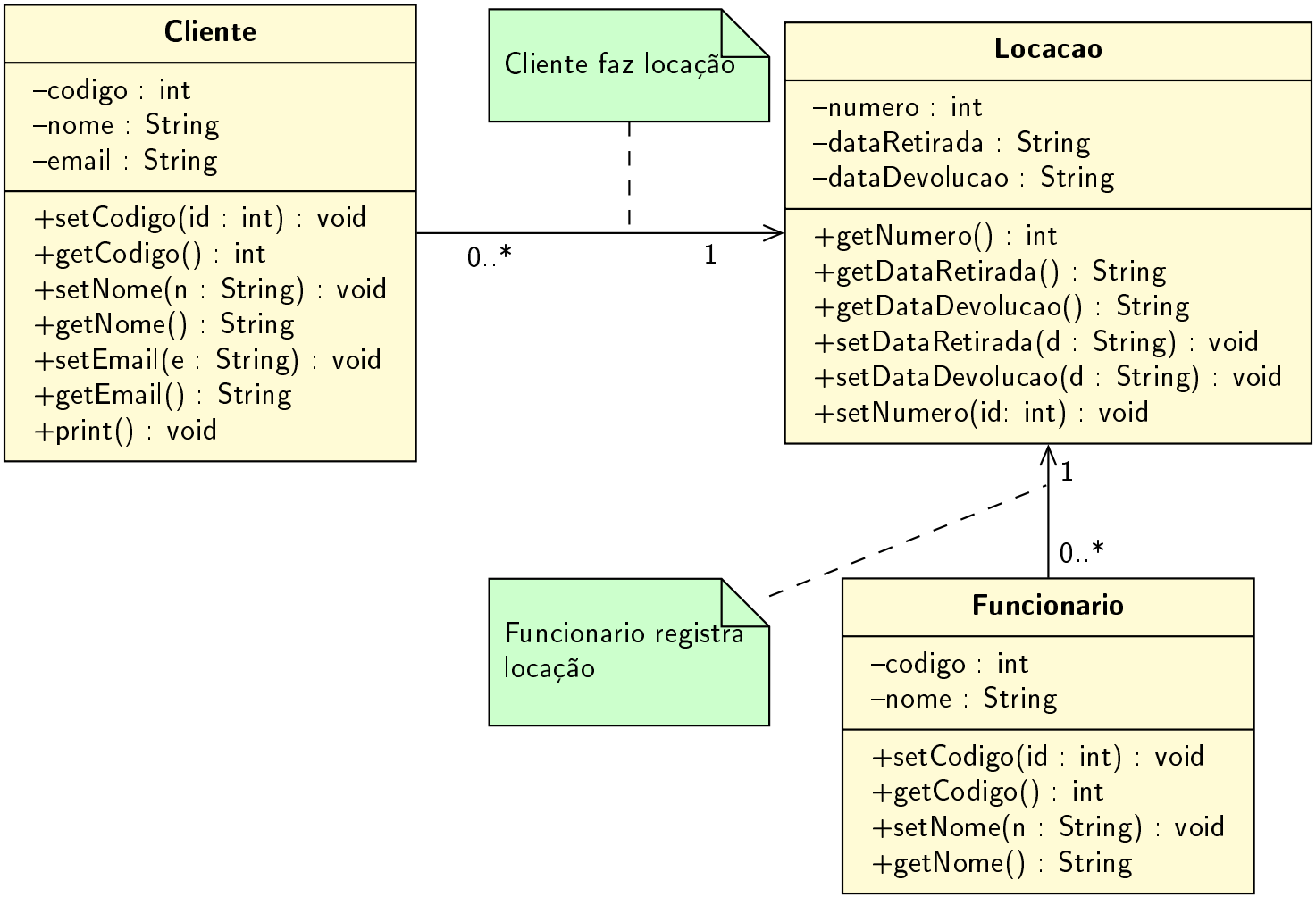
Associação simples III

Exemplos:

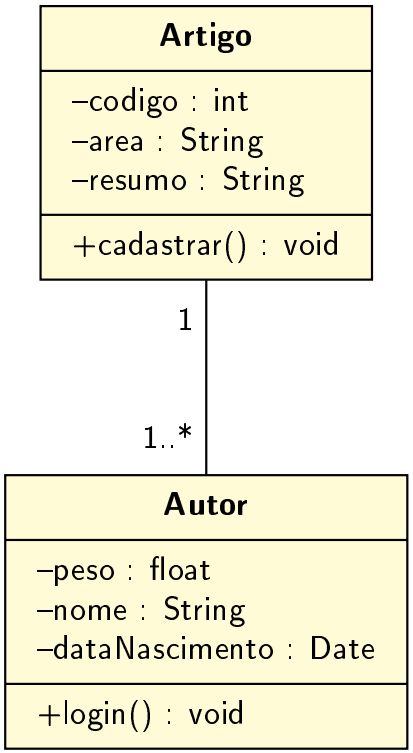
Associação simples IV



Associação simples V



Associação simples VI



Imagine um sistema de avaliação de artigos acadêmicos:

- Temos uma relação autor/artigo;
- Note que a classe Autor não compartilha atributos da classe Artigo e vice-versa;
- Nesse caso, não existe a relação *todo-parte*.

Essas duas formas de associação representam relacionamentos do tipo **"tem um"**

- Relacionamento **todo-parte** $\Rightarrow$ Uma classe é formada por, ou contém objetos de outras classes

Exemplos:

- Um carro contém portas;
- Uma árvore é composta de folhas, tronco, raízes;
- Um computador é composto de CPU, teclado, mouse, monitor, ...

Agregação e Composição II

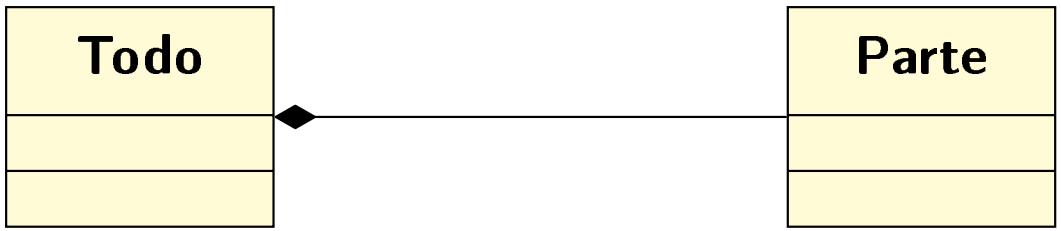
- **Composição**: relacionamento todo-parte em que as **partes** não podem existir independentes do **todo**:
  - Se o *todo* é destruído as *partes* são destruídas também;
  - Uma *parte* pode ser de um único *todo* por vez.
- **Agregação**: relacionamento todo-parte que não satisfaz um ou mais desses critérios:
  - A destruição do objeto *todo* não implica a destruição do objeto *parte*;
  - Um objeto pode ser *parte* componente de vários outros objetos.

Agregação × composição

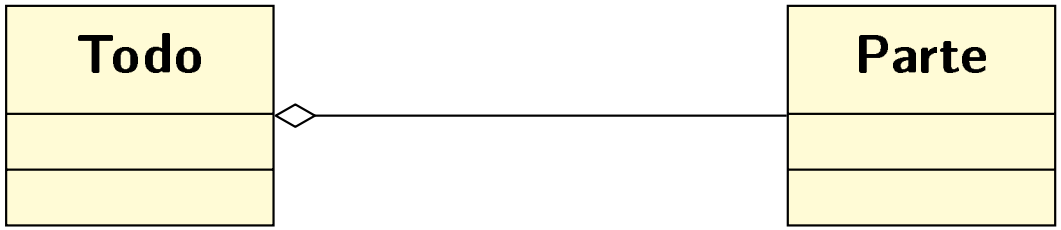
A **agregação** é uma forma mais fraca de **composição**.

Representação:

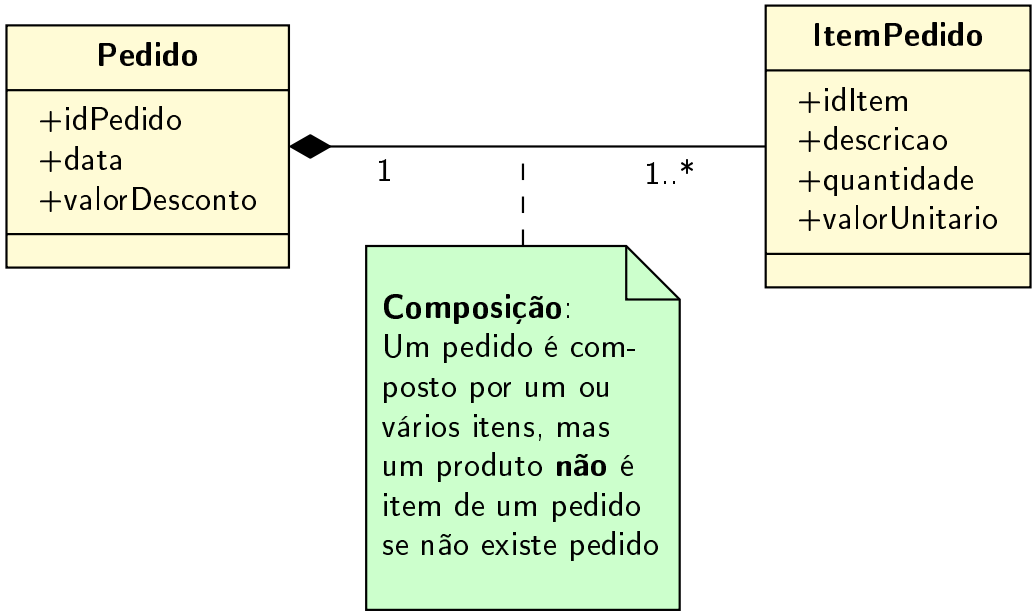
- **Composição**: associação representada com um losango **sólido** do lado *todo*.
  - O lado *todo* deve sempre ter multiplicidade 1.



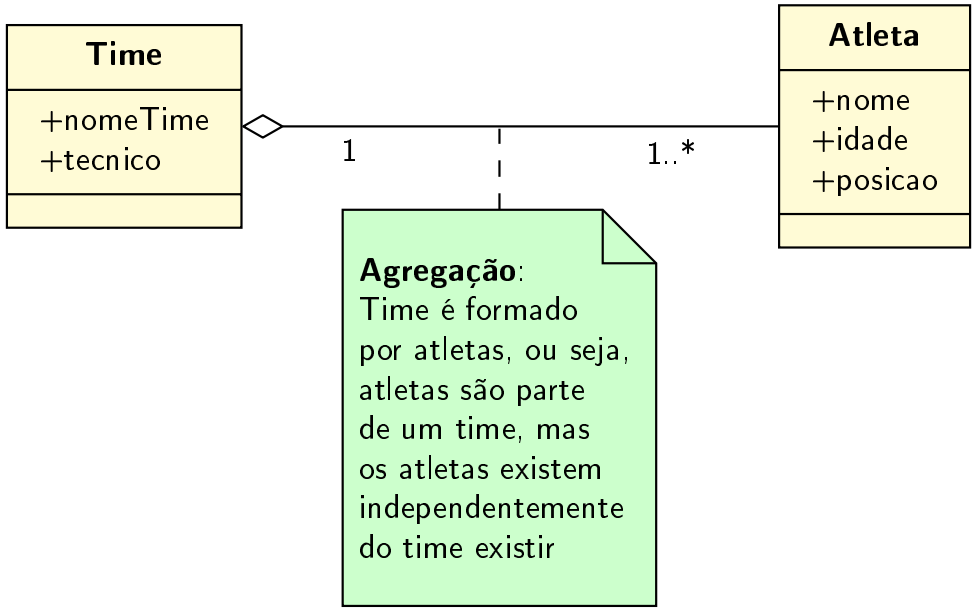
- **Agregação**: associação representada com um losango **sem preenchimento** do lado *todo*.



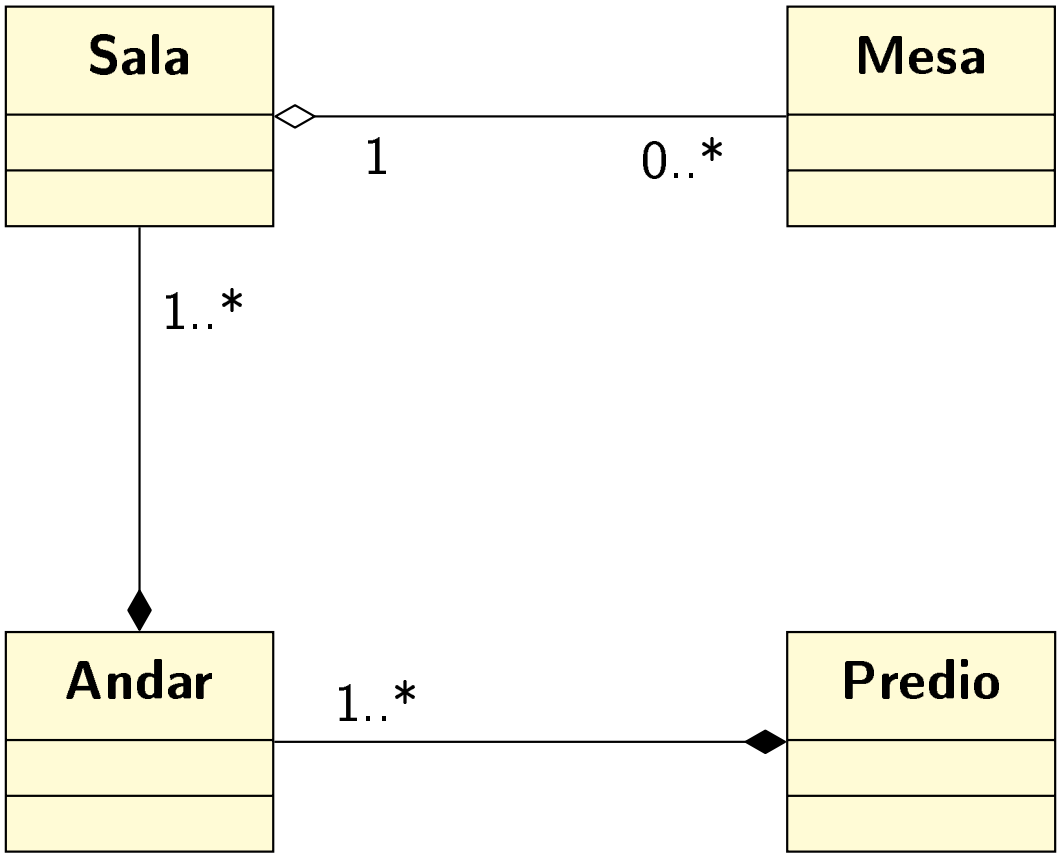
Exemplos:


Pedido
+idPedido
+data
+valorDesconto
1
1..*
ItemPedido
+idItem
+descricao
+quantidade
+valorUnitario
Composição:
Um pedido é composto por um ou vários itens, mas um produto não é item de um pedido se não existe pedido

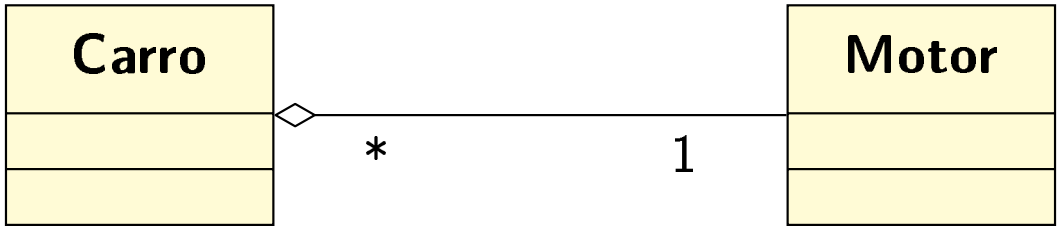
Agregação e Composição V



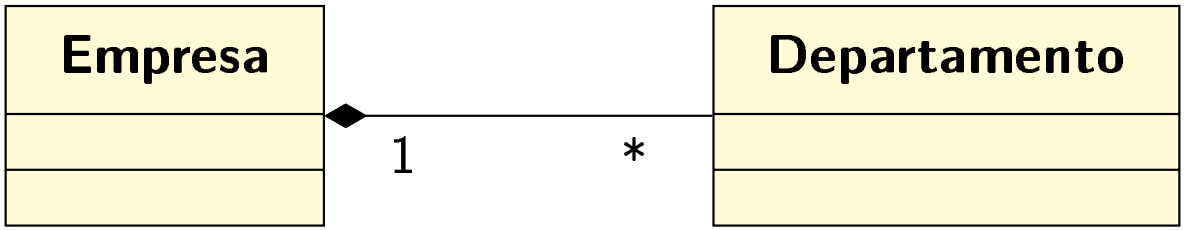
Agregação e Composição VI



Agregação:



Composição:



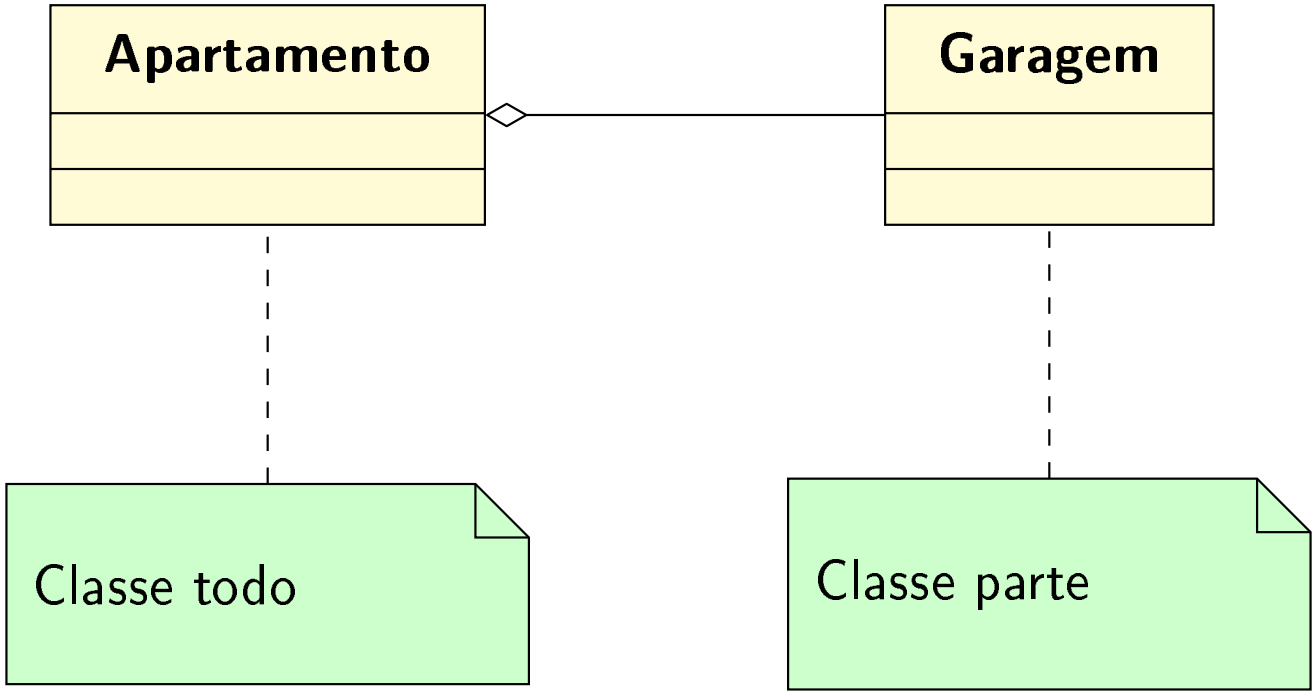
Classe todo:
É a classe resultante da agregação ou composição

Classe parte:
É a classe cujas instâncias formam a agregação/composição

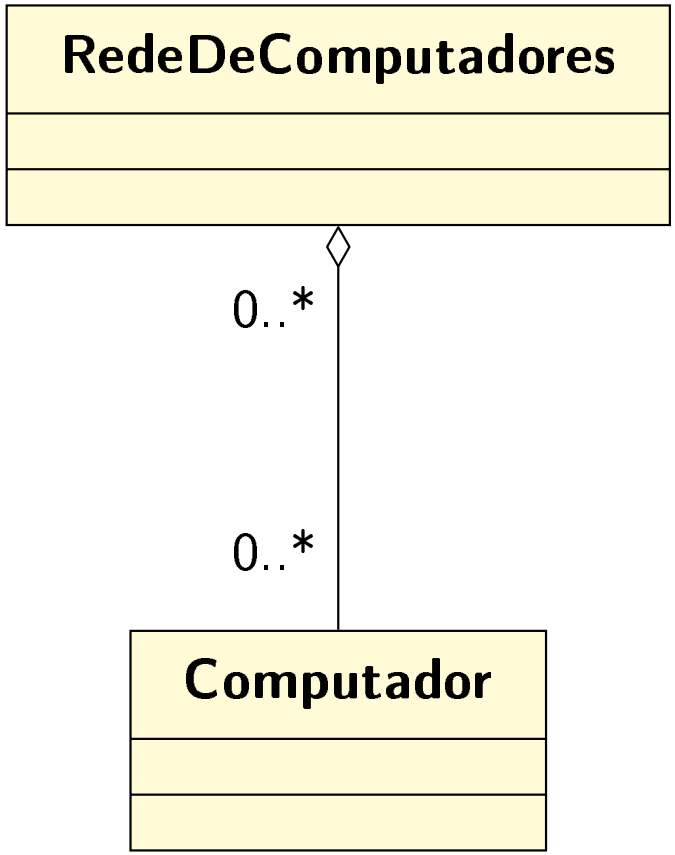
Exemplo:
Apartamento e Garagem: um apartamento pode ter garagem.

- Classe Apartamento: todo ou agregada
- Classe Garagem: parte

Agregação e Composição IX

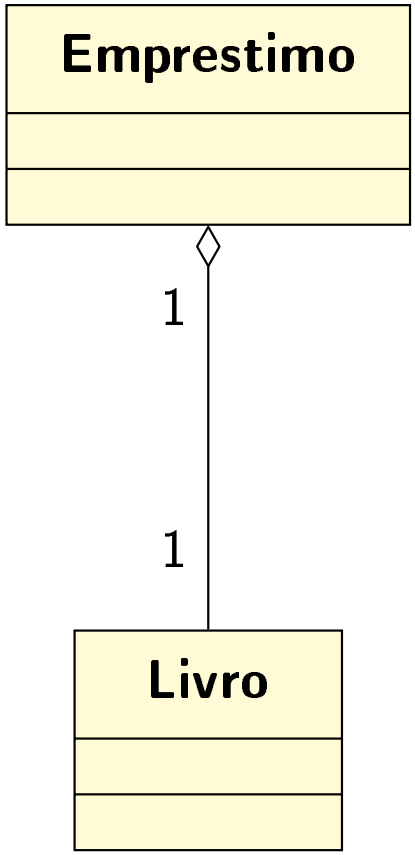


Agregação: mais exemplos I



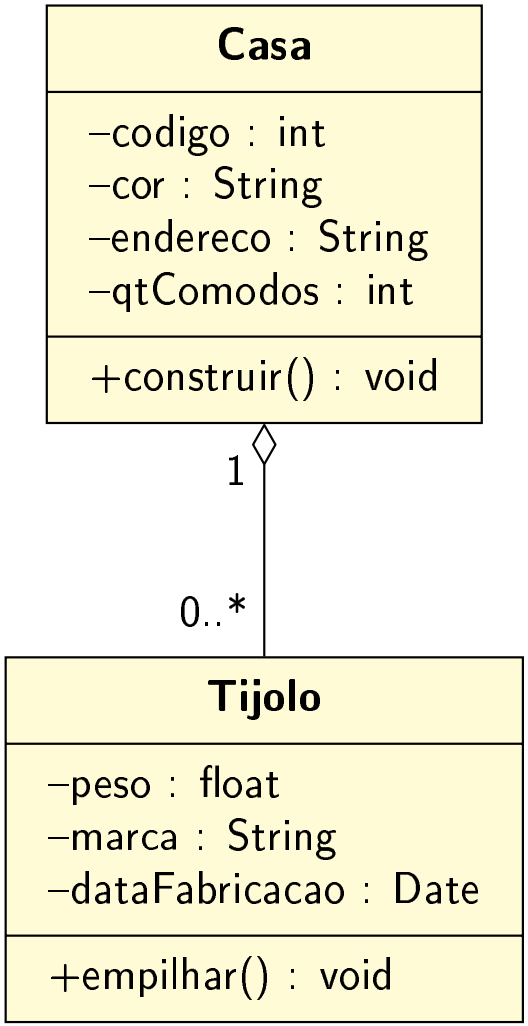
- Um computador existe independentemente de uma rede;
- Um computador pode estar ligado a mais de uma rede ao mesmo tempo.

Agregação: mais exemplos II



- Um empréstimo contém um livro, mas o livro não deixa de existir no sistema da biblioteca quando o empréstimo é concluído

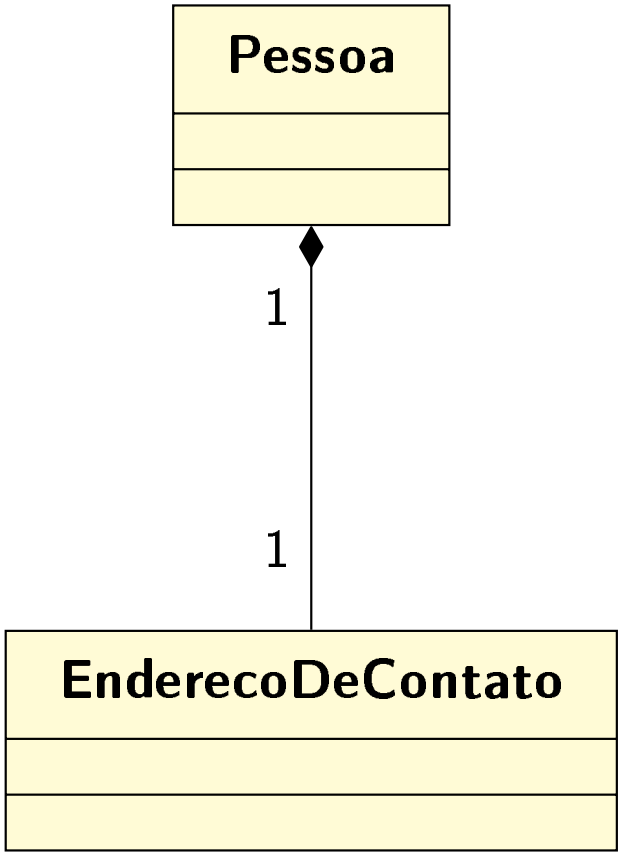
Agregação: mais exemplos III



Imagine um sistema de gerenciamento de obras e suponha as classes `Casa` e `Tijolo`:

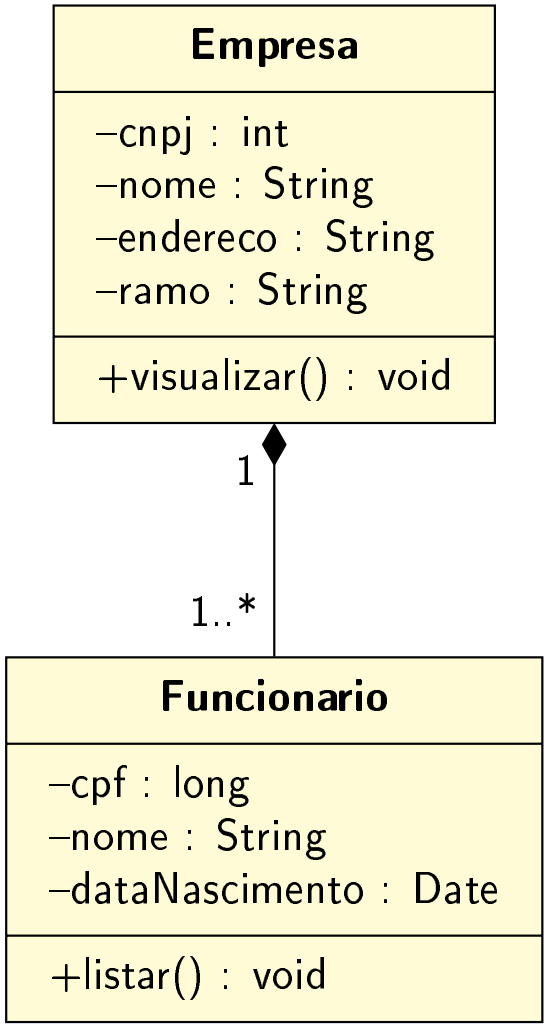
- Caso você deixe de construir uma casa, mesmo assim os tijolos poderão ser utilizados na construção de outro tipo de obra;
- Perceba que uma casa é feita de tijolos (relação **todo-parte**).

Composição: mais exemplos I



- O endereço de contato só faz sentido associado com uma pessoa;
- Se a pessoa é eliminada do sistema, não faz sentido manter o endereço de contato.

Composição: mais exemplos II



Imagine um sistema de Recursos Humanos e suponha as classes `Funcionário` e `Empresa`:

- Não faz sentido ter funcionários, se não existir uma empresa onde eles possam trabalhar;
- Se a empresa deixar de existir, automaticamente ela deixará de ter funcionários;
- Perceba que uma empresa é composta por funcionários (relação **todo-parte**).

Composição: mais exemplos III

```
class Pessoa {
    String nome;
    char sexo;
    Data dataNasc; // Data: classe
    ...
}
```

Composição: mais exemplos IV

```
class Data {
    private int dia, mes, ano;
    public void alteraData(int d, int m, int a){
        dia = d;
        mes = m;
        ano = a;
    }
}
```

Relacionamentos: qual usar?

Como saber qual relacionamento deve ser utilizado?

- Existem atributos ou métodos sendo aproveitados por outras classes? A subclasse "é do tipo" da superclasse?
- **Sim**: Isso é herança
- **Não**: Existe todo-parte?
  - **Sim**: A parte vive sem o todo?
    - ★ **Sim**: Isso é agregação
    - ★ **Não**: Isso é uma composição
  - **Não**: Isso é associação

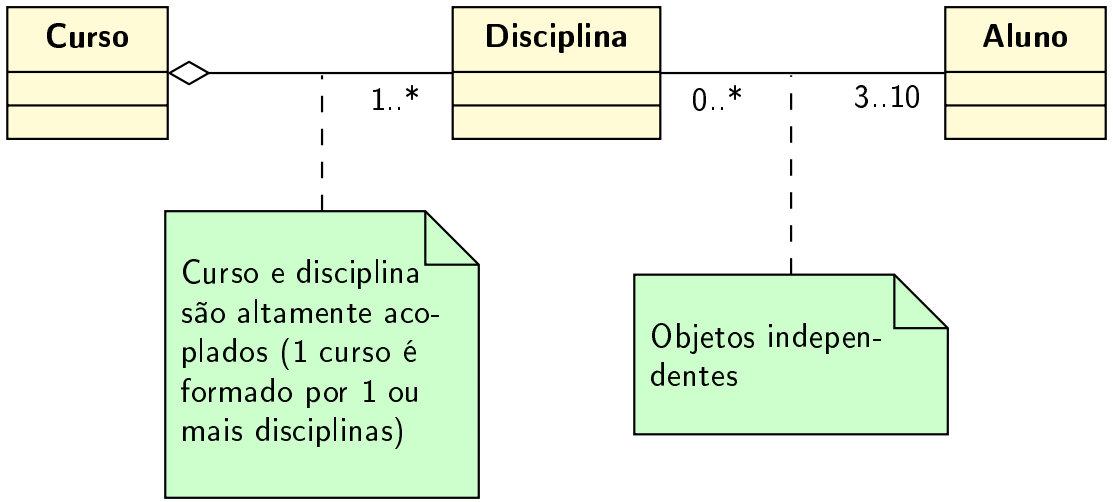
Associação vs. agregação/composição I

Se dois objetos são altamente acoplados por um relacionamento todo-parte:

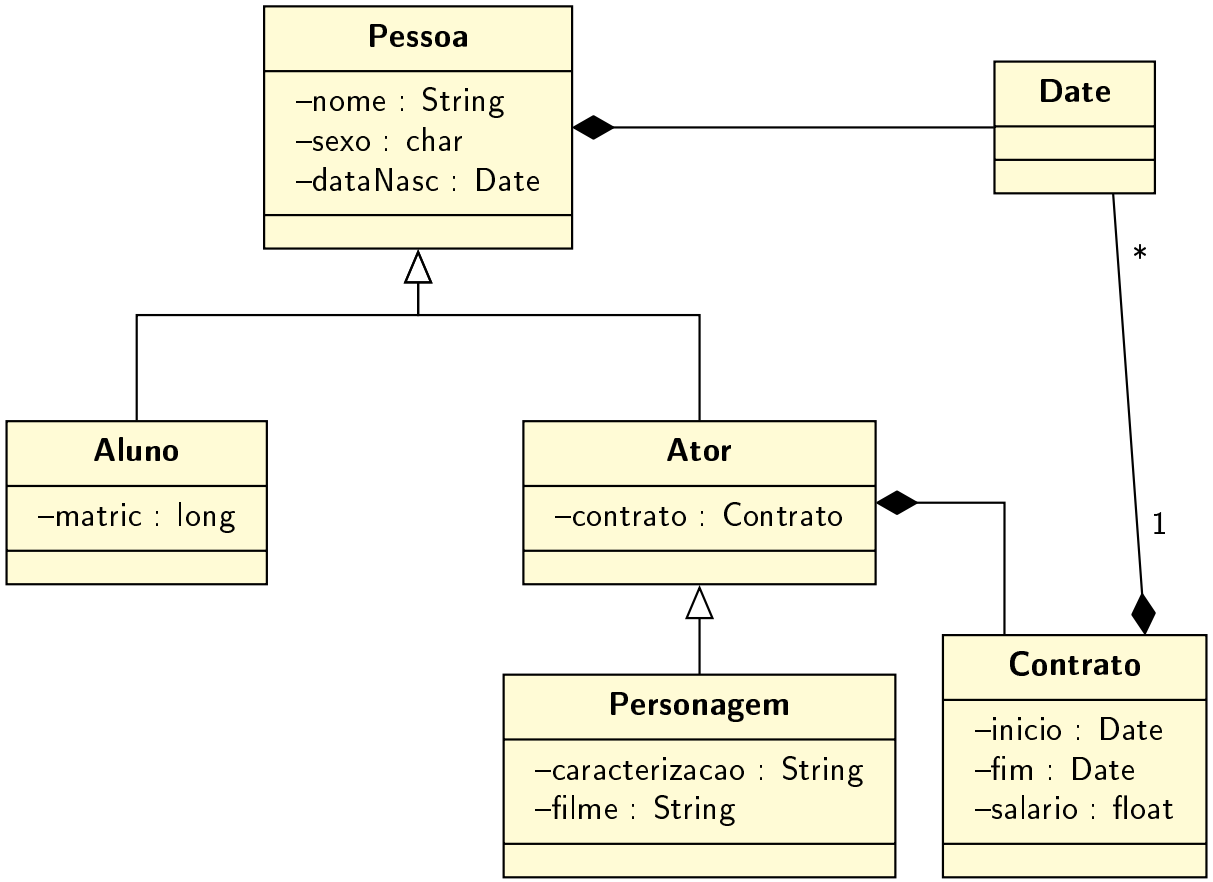
- O relacionamento é uma **agregação** ou **composição**.

Se dois objetos são usualmente considerados como independentes, mesmo que eles estejam frequentemente ligados:

- O relacionamento é uma **associação**.



Exemplo:


Pessoa
–nome : String
–sexo : char
–dataNasc : Date
Date
*
1
Aluno
–matric : long
Ator
–contrato : Contrato
Personagem
–caracterizacao : String
–filme : String
Contrato
–inicio : Date
–fim : Date
–salario : float

Classes de associação I

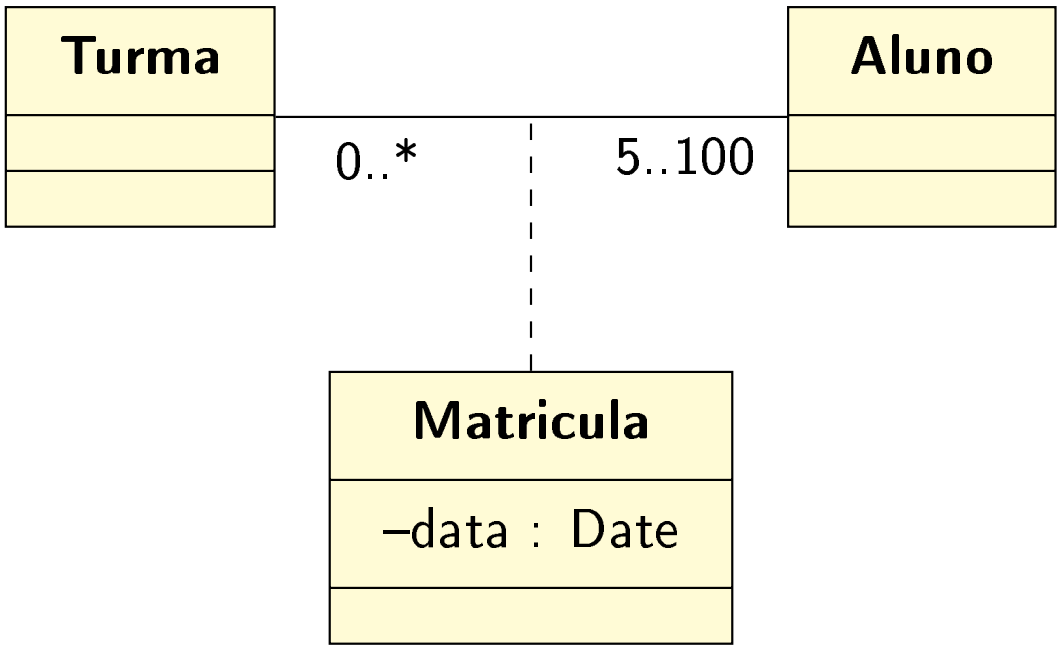
As **classes de associação** são classes que fornecem um meio para adicionar atributos e operações a associações.

- Normalmente geradas entre ocorrências que possuem multiplicidade muitos nas extremidades
- **Exemplo**, considere o relacionamento a seguir:

- Deseja-se agora acrescentar a data em que cada aluno foi adicionado à turma;

Classes de associação II

- Obviamente, esta data não é uma propriedade *nem do aluno* e *nem da turma*.
- Sendo assim, criamos uma **classe associativa**, chamada, por exemplo, de Matricula:



Exercícios I

1. Faça a modelagem em UML de um sistema bancário, relacionado à administração de contas bancárias (para cada classe defina, pelo menos, 4 atributos e 4 métodos). Em um banco há gerentes responsáveis por um grupo de clientes.
   - Um gerente poderá aceitar pedidos de abertura de conta, de empréstimo, de cartão de crédito, etc. Mas poderá decidir por oferecer os serviços, ou não.
   - Cada cliente poderá pedir serviços para um gerente: abertura de contas, empréstimo, cartão de crédito, etc. Ele também poderá ter acesso à sua conta bancária.
   - Cada conta bancária poderá oferecer serviços tais como: depositar, sacar, transferir dinheiro entre contas, pagar cartão de crédito, etc.
   - Após a modelagem, para cada classe coloque quais serviços pode solicitar das outras classes.

Exercícios II

Gerente
–nome : String –funcao : String –numeroClientes : int –cpf : long
+iniciarPedidoEmprestimo() +iniciarPedidoCartao() –liberarEmprestimo() –liberarCartao()

Cliente
–nome : String –cpf : long –salario : double –profissao : String
+atualizarSenha() +cadastrarComputador() +pedirEmprestimo() +pedirCartao()

ContaBancaria
–nomeCliente : String –tipoConta : String –validade : String –dataCriacao : String
+depositar() +sacar() +transferir() +pagarCartao()

Exercícios III

2. Faça a modelagem em UML de um sistema de controle de cursos de informática equivalente ao módulo de matrícula de acordo com os seguintes fatos:
   - o curso pode ter mais de uma turma, no entanto, uma turma se relaciona exclusivamente com um único curso.
   - uma turma pode ter diversos alunos matriculados, no entanto uma matrícula refere-se exclusivamente a uma determinada turma. Cada turma tem um número mínimo de matriculas para iniciar o curso.
   - um aluno pode realizar muitas matrículas, mas cada matrícula refere-se exclusivamente a uma turma específica e a um único aluno.

Exercícios IV

3. Faça a modelagem em UML de um sistema de reserva para uma empresa aérea (para cada classe defina, pelo menos, 4 atributos e 4 métodos).
   - Cada voo deverá estar cadastrado no sistema, pois as reservas serão relacionadas a eles. Cada voo pode informar o número de assentos livres, sua tripulação, reservar acento, etc
   - Operadores são funcionários da empresa responsáveis pela operacionalização das reservas. Os operadores fazem as reservas, as cancelam, informam sobre possíveis atrasos, etc
   - Os clientes podem pedir reservas nos voos, podem cancelar reservas, podem pagá-las de forma adiantada, etc

Após a modelagem, para cada classe coloque quais serviços pode solicitar das outras classes.

Referências

1. Addison-BOOCH, G., RUMBAUGH, J., JACOBSON, I. *UML, Guia do Usuário*. Rio de Janeiro: Campus, 2000.
2. FOWLER, M. *UML Essencial*, 2a Edição. Porto Alegre: Bookman, 2000.
3. LARMAN, C. *Utilizando UML e Padrões: Uma Introdução à Análise e ao Projeto Orientado a Objetos*. Porto Alegre: Bookman, 2001.

Os slides dessa apresentação foram cedidos por:

- Graça Marietto e Francisco Zampirolli, UFABC
- Profa Katti Faceli, UFSCar/Sorocaba
- Marcelo Z. do Nascimento, FACOM/UFU

LaTeXagem: Renato Pimentel, FACOM/UFU