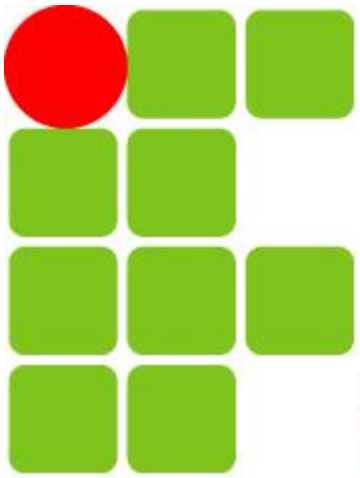
Modelagem Conceitual Estrutural – Parte 2

Análise de Sistemas



INSTITUTO FEDERAL
ESPÍRITO SANTO

Ligações e associações

- Objetos relacionam-se entre si:
  - Ligação: conexão entre objetos;
  - Associação: conexão entre classes que representa a existência de ligações;
  - Uma associação descreve um conjunto de potenciais ligações da mesma maneira que uma classe descreve um conjunto de potenciais objetos [Rumbaugh].
  - **Modelamos Classes e Associações!**

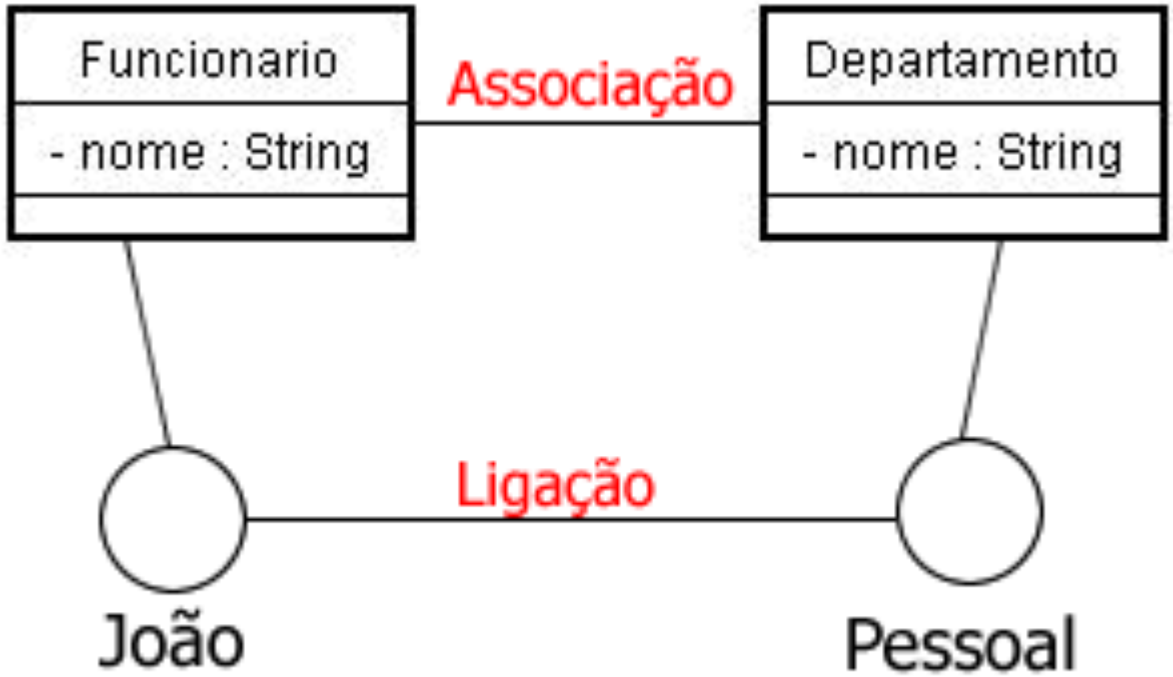
Habitantes

Aná

Cão de Guarda

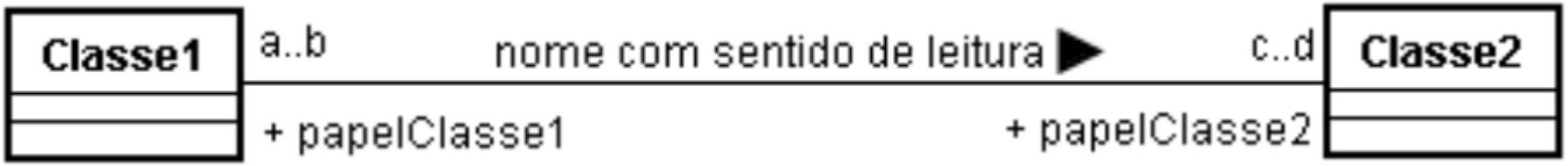
Classe: Pessoa **Classe: Casa** **Classe: Cachorro**

Ligações e associações



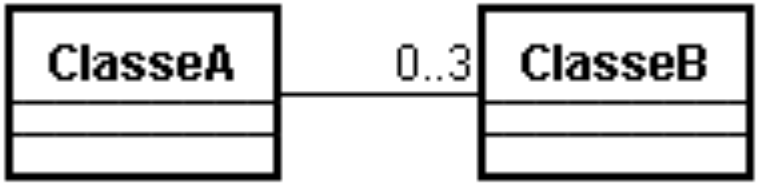
Associações

- Podem ser nomeadas
  - Uso de verbos conjugados, indicando o sentido de leitura
  - Sentido de leitura é diferente de navegabilidade! Em modelos conceituais as associações são navegáveis nos dois sentidos!
- Papeis indicam o papel de cada classe na associação
- Cardinalidades ou multiplicidades

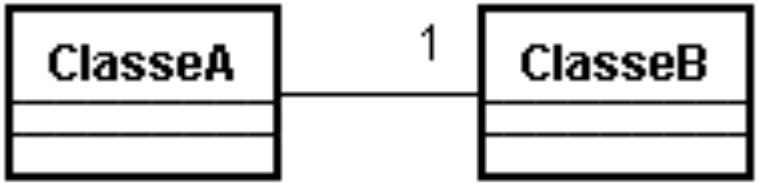


Cardinalidades

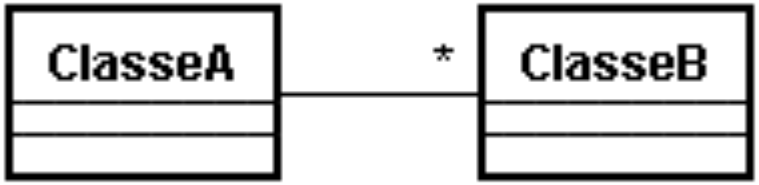
- Indicam quantos objetos podem participar de um determinado relacionamento.



Objetos da ClasseA podem se relacionar com no mínimo zero e no máximo três objetos da ClasseB.



Um e somente um.



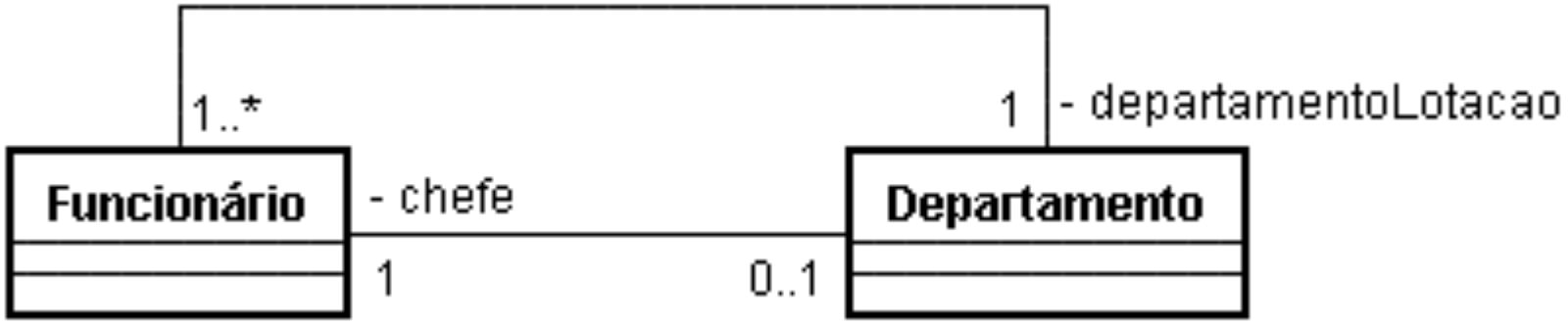
Nenhum, um, ou vários.

Cardinalidades

- Voltando ao exemplo da biblioteca
  - Como representar que cada usuário pode fazer até três reservas e cada reserva é de apenas um usuário?
  - Como representar que cada reserva é feita para reservar um livro, mas cada livro pode estar envolvido em várias reservas?

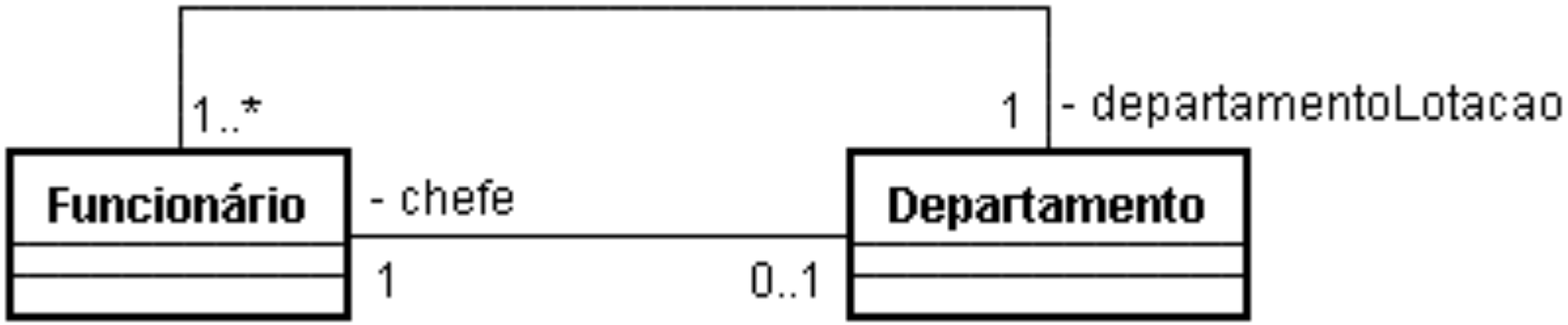
Papéis

- Indicam o papel que a classe desempenha na associação (são usados substantivos);
- É opcional, usado quando melhora o entendimento do modelo;
- Sintaxe: <escopo> <nome>.



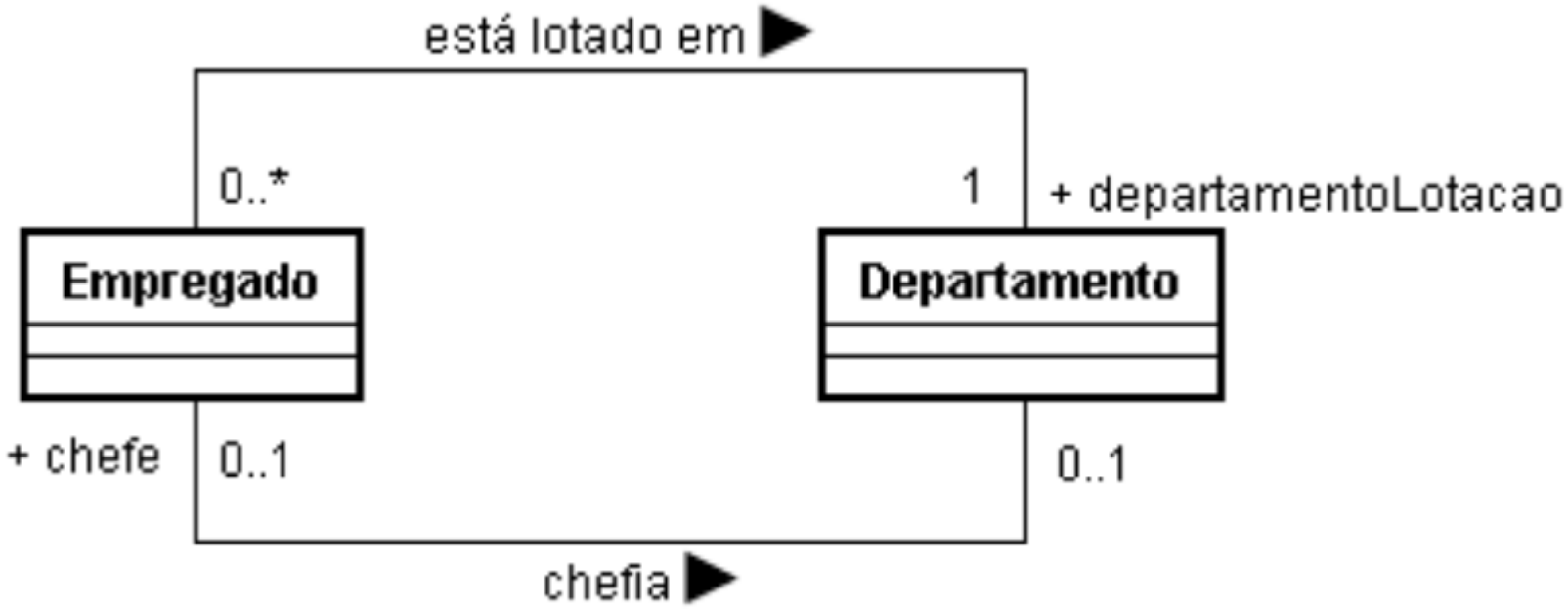
Importante!!!

- Não representa-se atributos para mapear associações.
- O "chefe" do departamento está representado no diagrama abaixo por uma associação, pois seria um atributo do tipo "Funcionario"
- A representação de associações como atributos será vista em maiores detalhes na disciplina de Projeto!



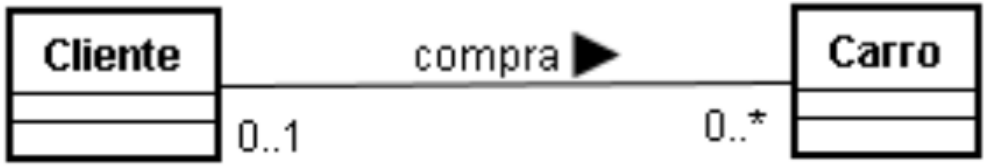
Identificação de associações

- Pode haver mais de uma associação entre duas classes!
  - Nestes casos é especialmente importante nomear associações e papéis.

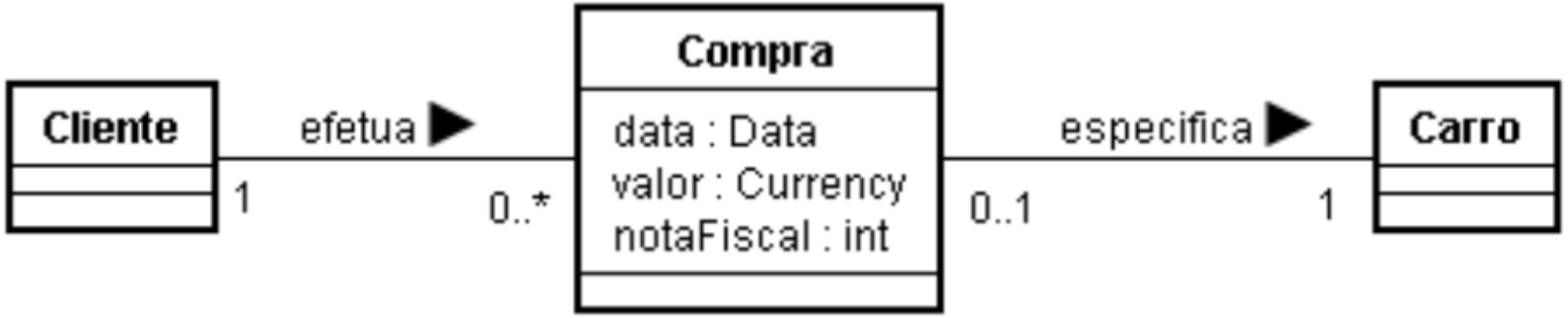


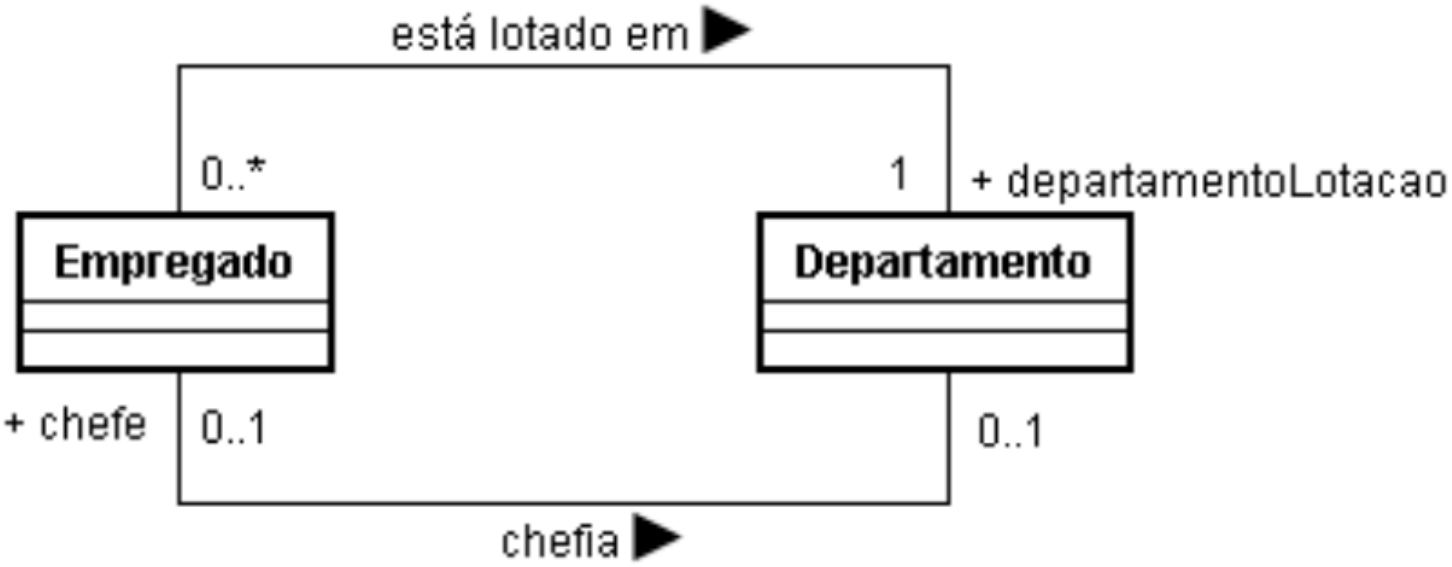
Eventos como classes ou associações

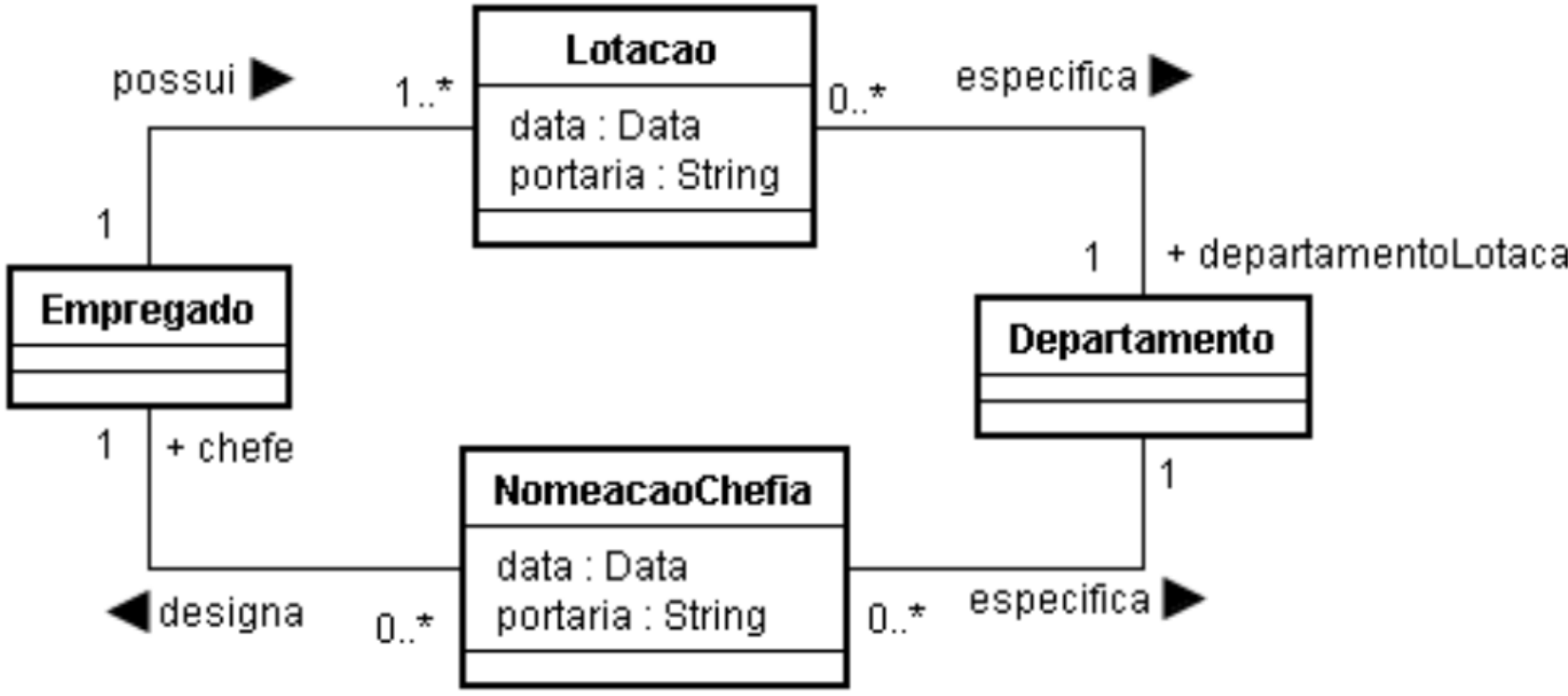
(a)



(b)

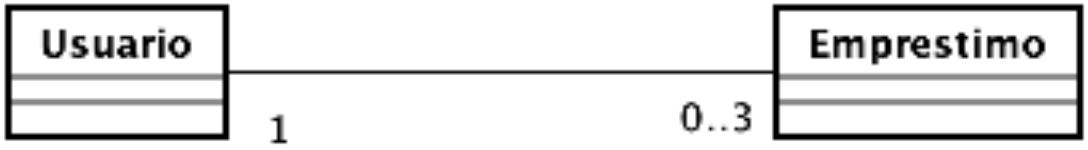



está lotado em
0..*
1
+ departamentoLotacao
Empregado
Departamento
+ chefe
0..1
0..1
chefia

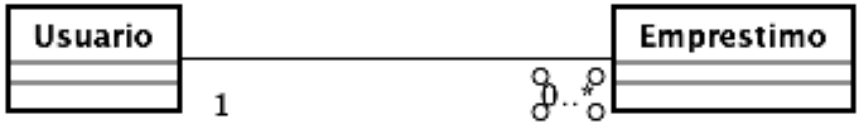

possui
1..*
Lotacao
data : Data
portaria : String
0..*
especifica
1
1
+ departamentoLotaca
Empregado
Departamento
1
+ chefe
1
NomeacaoChefia
data : Data
portaria : String
designa
0..*
0..*
especifica

Relações históricas

- Um usuário só pode ter 3 empréstimos em andamento.

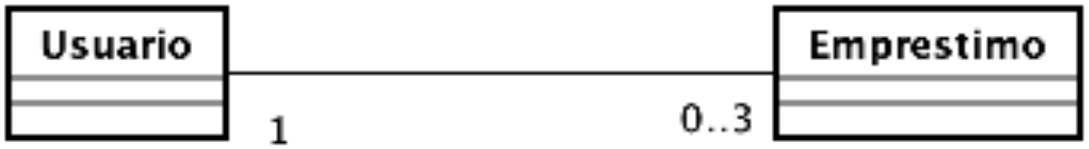


- Mas não há limites para empréstimos durante sua existência.

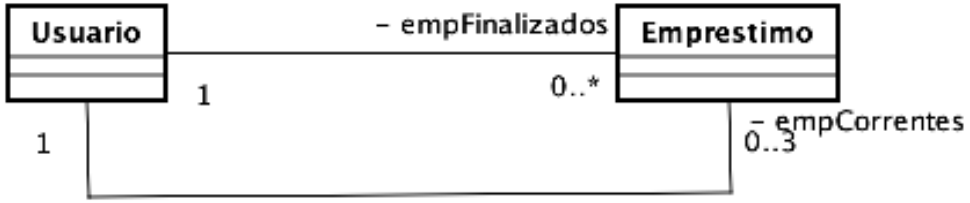


Relações históricas

- Como resolver passa por uma questão: quero/preciso armazenar o histórico de empréstimos?
- Se a resposta for não...

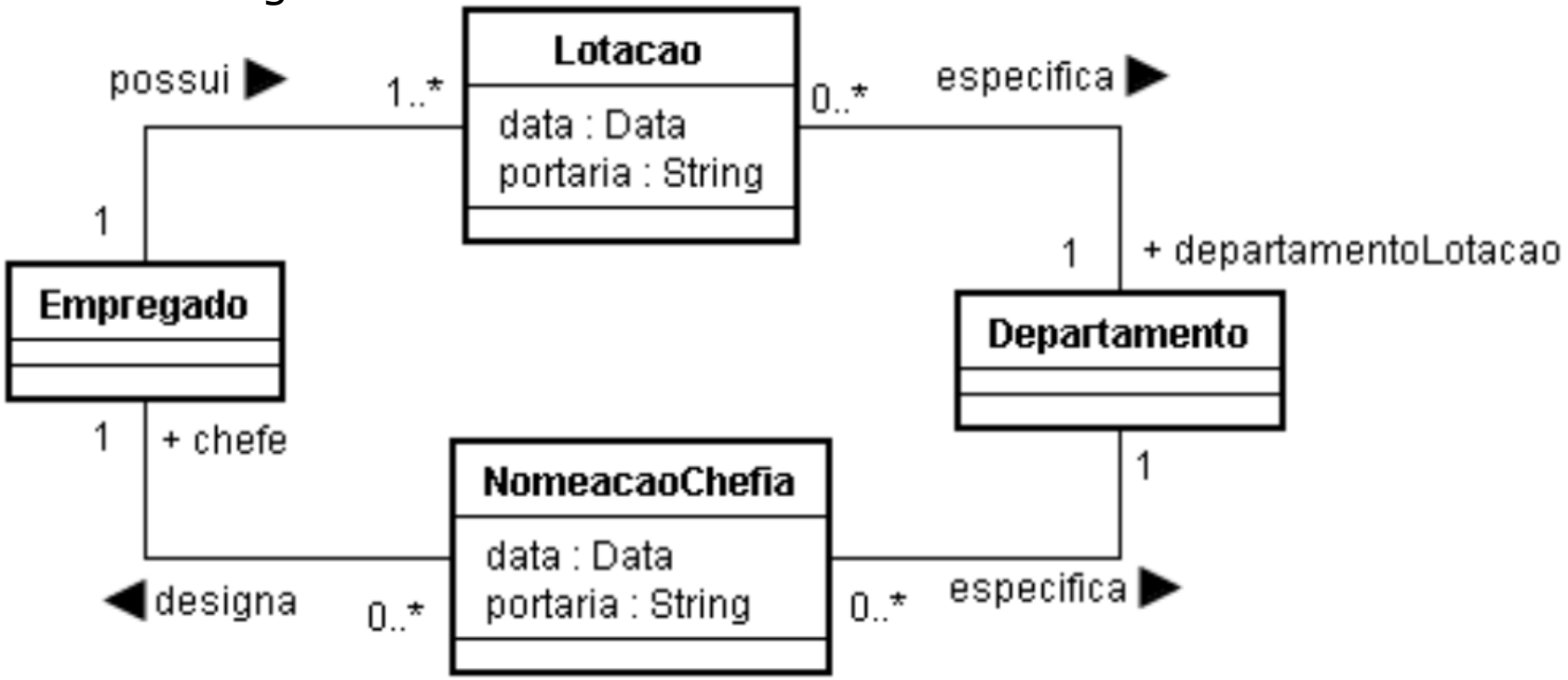


- Se for sim...



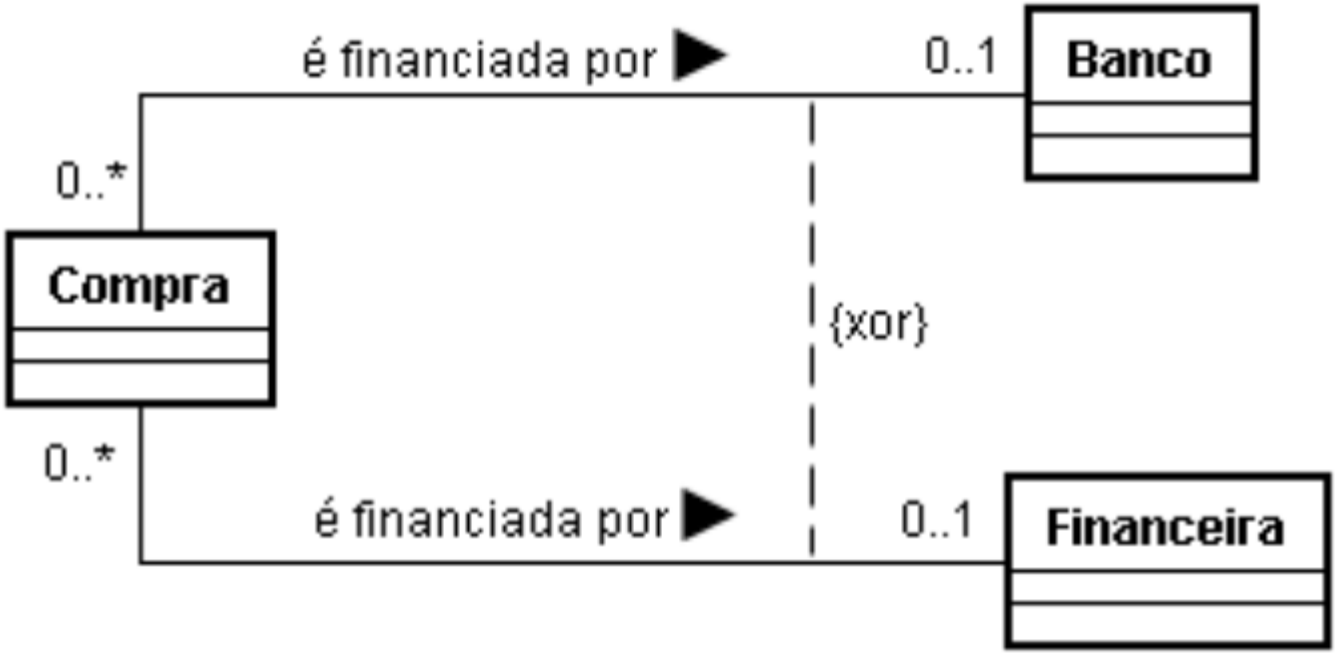
Restrições de Integridade

Algumas restrições não expressas graficamente. Ex.:
Um empregado só pode estar lotado em um departamento em um momento.
Um departamento só pode ter um chefe em um momento.
Um departamento só pode ter como chefe um empregado lotado nele.
Data da nomeação de Chefia deve ser maior que a data de lotação...
Essas Regras Devem ser descritas no modelo conceitual!!!



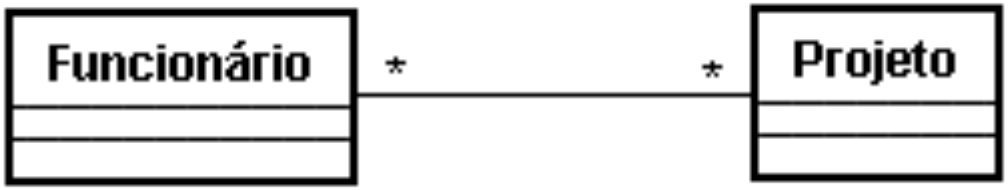
Restrições de Integridade

Recurso para representar restrições graficamente:
-Multiplicidades
-Restrições entre {}



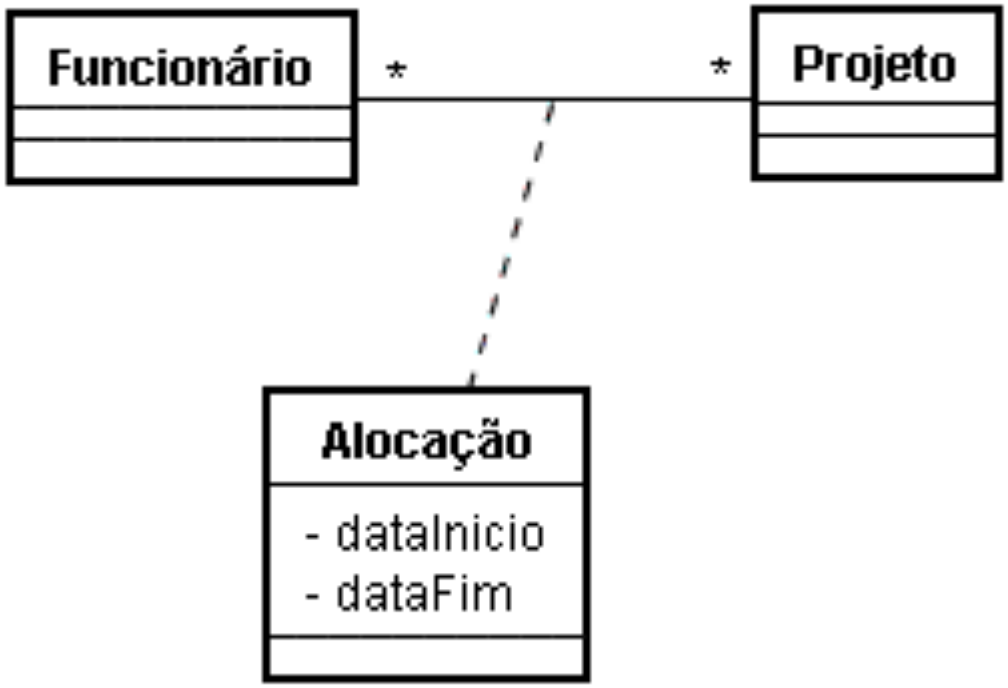
Associações muitos-para-muitos

- Perfeitamente legais;
- Requerem atenção à possíveis atributos do relacionamento (eventos a serem lembrados).



É requisito do sistema armazenar data de início e fim da alocação de um funcionário a um projeto?

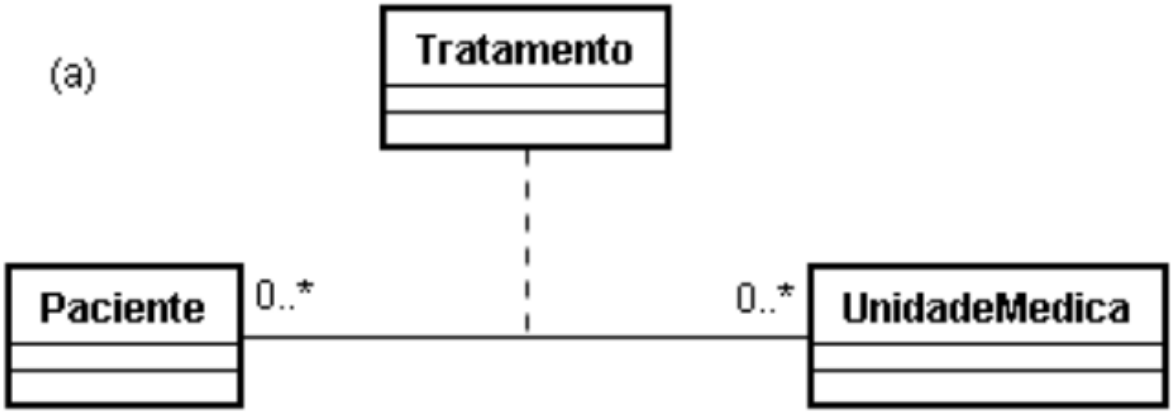
Reificando associações com Classes associativas



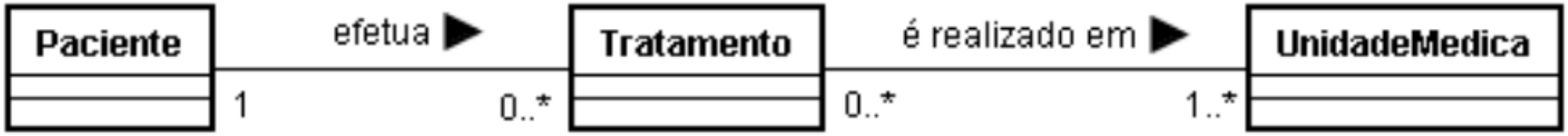
Reificação de Associações

Os modelos abaixo tem semânticas diferentes devido à multiplicidade Definida em UnidadeMédica!!!.

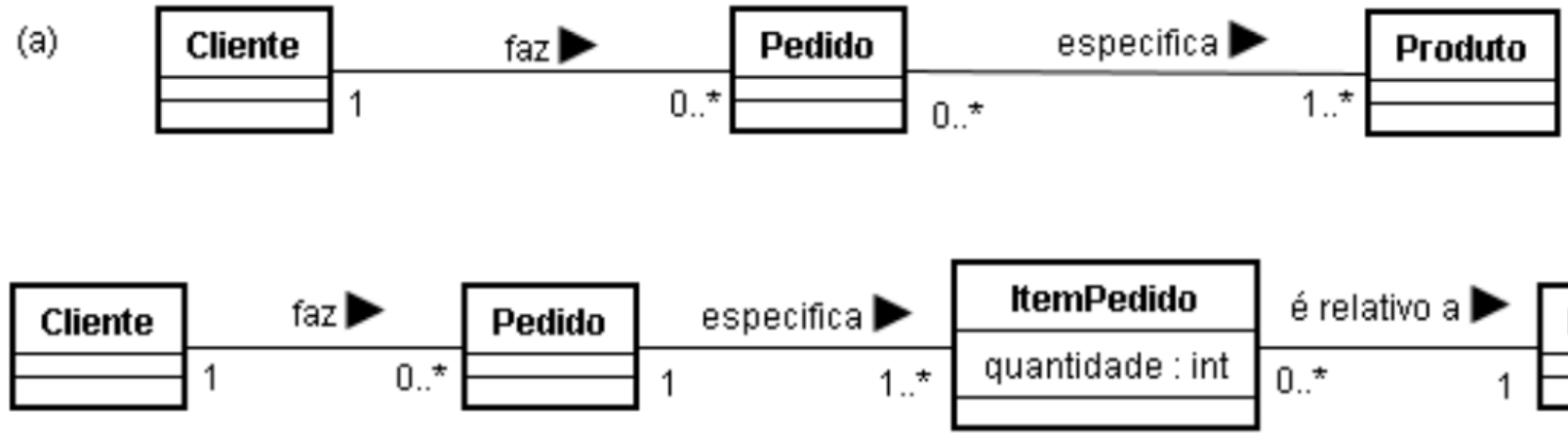
(a)



(b)

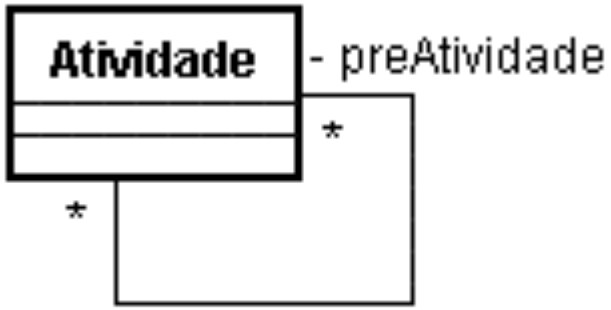


Reificação de Associações



Relacionamentos recursivos

- Perfeitamente legais;
- Geralmente pedem definição de papéis.



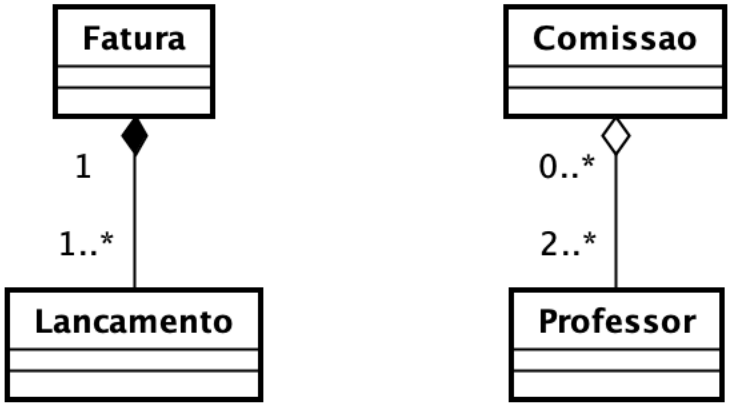
Relações todo-parte: Composição e agregação

- Na agregação não há uma dependência existencial entre "todo" e "parte"

- A composição, é mais forte! Nela há uma dependência existencial entre a parte e o todo.
  - Como consequência, sempre que o todo for excluído, as partes também serão!

Composição e agregação

- UML: Agregação representada por um losango branco e composição por um losango preto.

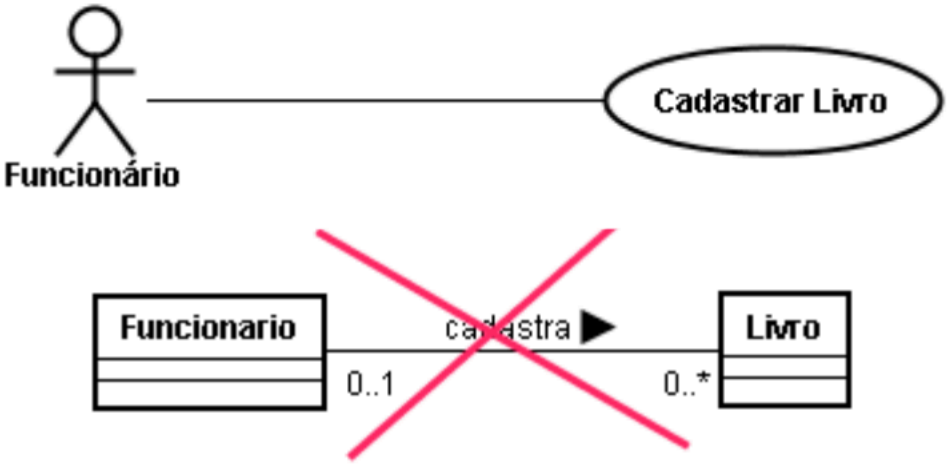


Composição e agregação

- Pode ser difícil perceber que uma relação trata-se de uma todo-parte. Na dúvida use associação comum por impor menos restrições.

Casos de uso e relações

- Às vezes há relações entre entidades que não são importantes para o sistema em questão...

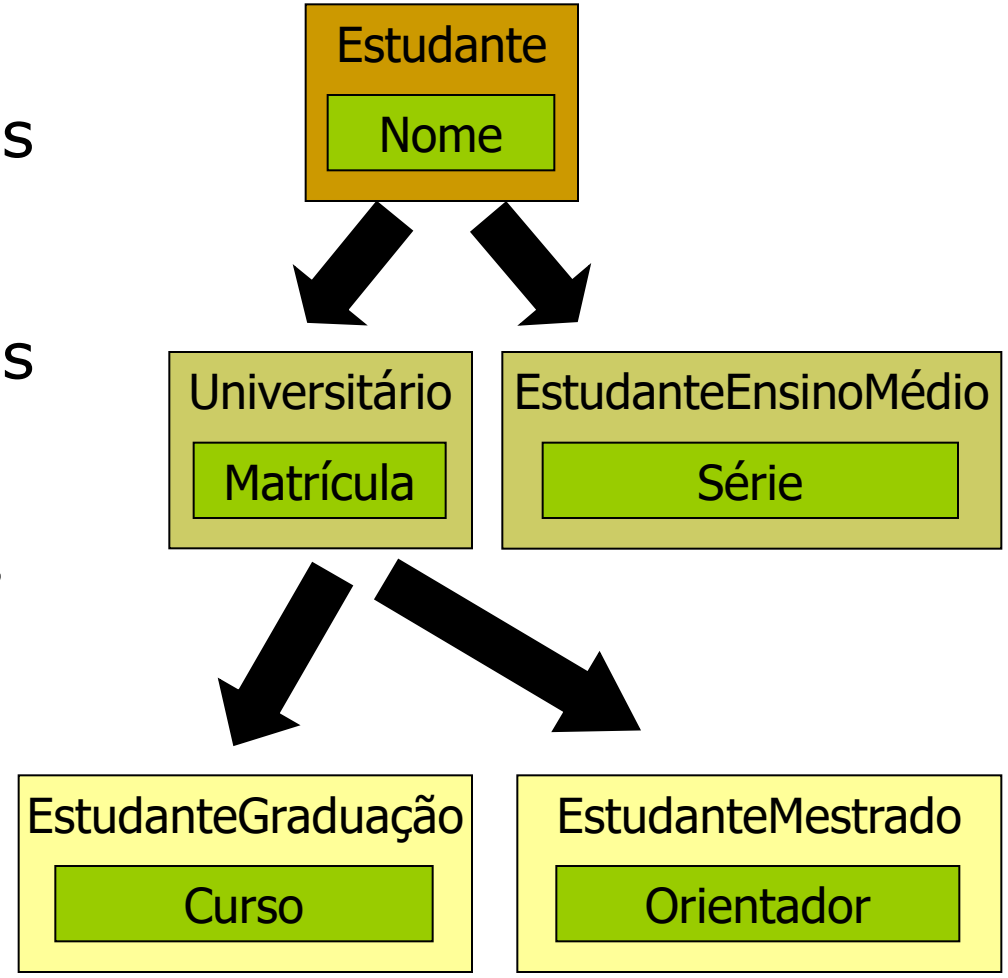


Generalização/especialização

- Mecanismo também conhecido como herança
- Generalização e especialização são úteis para estruturação do sistema;
- São formadas hierarquias de classes:
  - "Filhos" (ou subclasses) herdam estrutura e comportamento dos "pais" (ou superclasses) e demais "ancestrais", de forma indireta.
- A herança possibilita:
  - Reutilização;
  - Captura explícita de características comuns;
  - Definição incremental de classes.

Generalização/especialização

- Generalização: quando classes têm semelhanças podemos definir uma classe mais geral;
- Especialização: muitas vezes um conceito pode ser refinado, adicionando-se novas características.
- **Só depende da direção de leitura!**



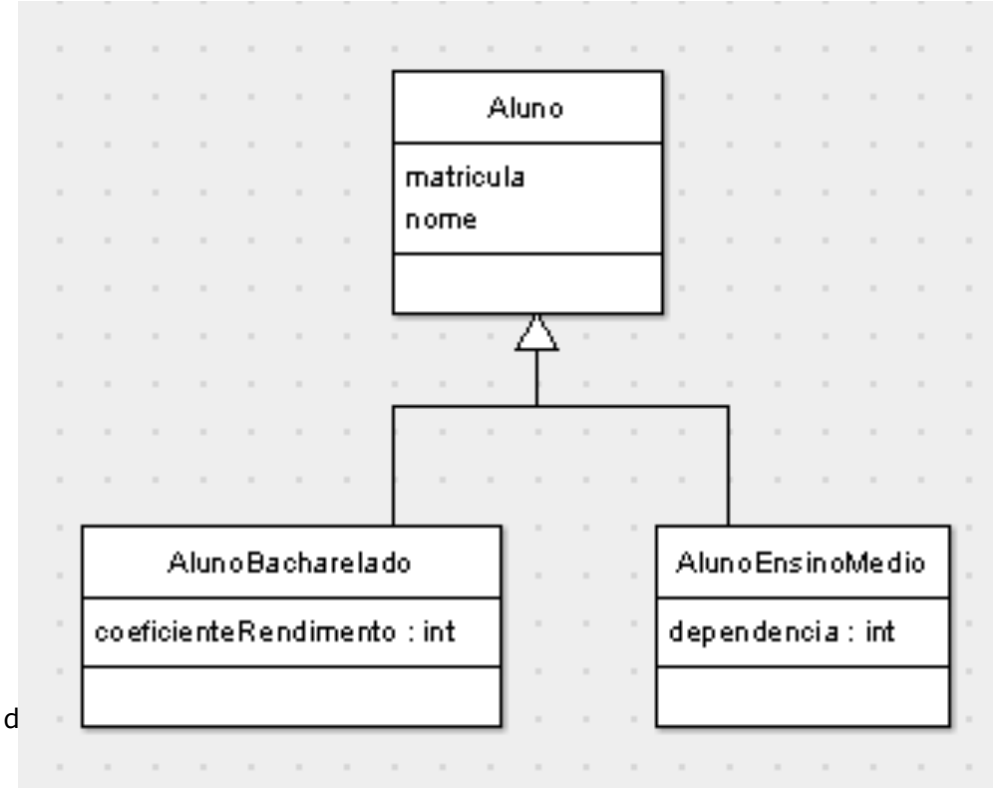
Cuidados com o uso de herança

- Semântica da herança:
  - "é um tipo de";
  - "é uma instância indireta de";
  - Ex.: Universitário é um tipo de Estudante.
- Semântica diferente das associações:
  - associações mapeiam ligações entre instancias das classes enquanto
  - herança é uma relação entre classes que indica que as instancias da subclasse são também instâncias da superclasse

Cuidados com o uso de herança

- Quando um conjunto de classes possuir semelhanças e diferenças elas podem ser organizadas numa hierarquia de classes utilizando heranças.

- Se a subclasse precisa cancelar características da superclasse, há algo errado!



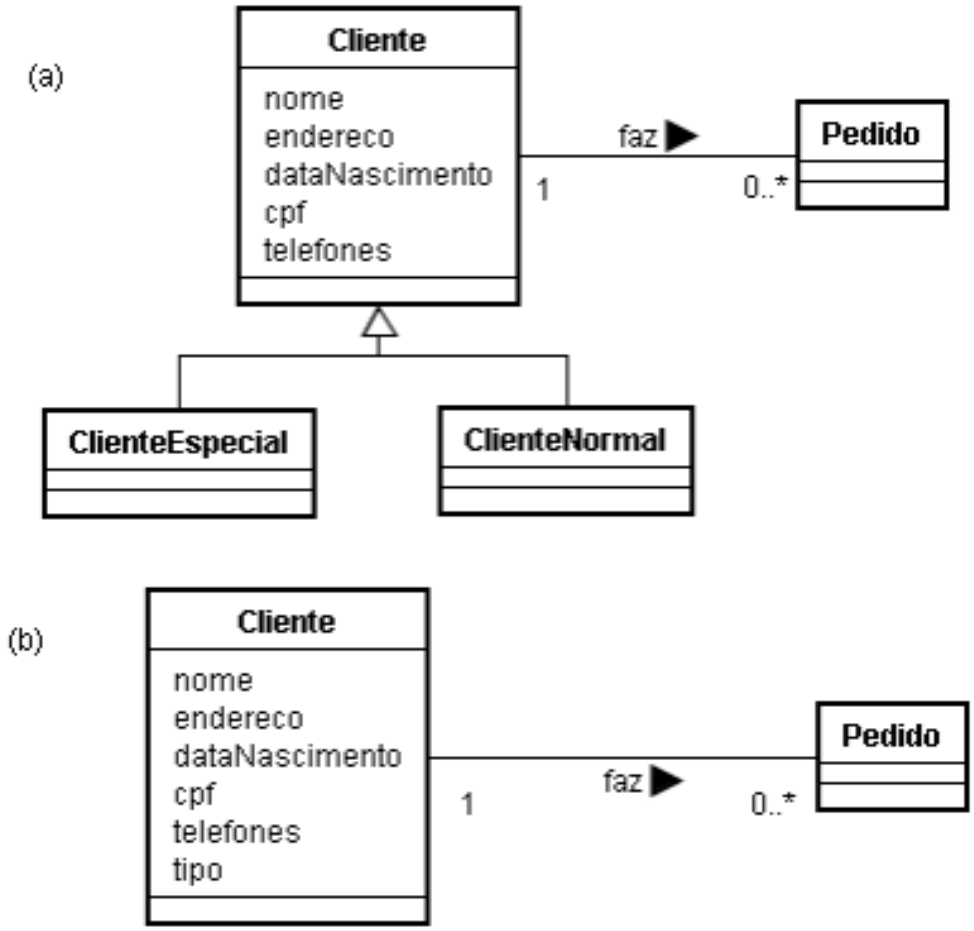
Diretrizes para herança

- Deve modelar relações “é-um-tipo-de”;
  - Instâncias da subclasse são por definição instâncias da superclasse. Tudo que é dito para esta última vale para a primeira.
- Subclasses devem suportar toda a funcionalidade das superclasses e possivelmente mais;
- Funcionalidade comum a diversas classes deve estar o mais alto possível na hierarquia;

Diretrizes para herança

- Classes abstratas não podem herdar de classes concretas;
- A herança deve estar no domínio do problema e fazer parte das responsabilidades do sistema;
- Classes que não adicionam funcionalidade nem redefinem funcionalidades existentes podem ser eliminadas (podem ser substituídas pelo uso de atributos de tipos enumerados) ***(exemplo no próximo slide).***

Diretrizes para herança



Conceitos avançados em Herança

- Herança Múltipla: ocorre quando uma classe herda de mais de uma outra.
  - Não é suportado em algumas linguagens, como Java
- Hierarquias ortogonais são perfeitamente possíveis.
  - Geram classificação múltipla: um objeto é instância de várias classes sendo que não há herança entre elas

Conceitos avançados em Herança

- Generalization Set: conjunto de generalizações que utilizam o mesmo critério para criar subclasses

- Constraints de um Generalization Set :
  - Disjoint ou overlapping
  - Complete ou incomplete
  - Representado entre chaves ({})
  - O padrão é {incomplete, disjoint}

Conceitos avançados de OO

Classes Abstratas

Operações Abstratas

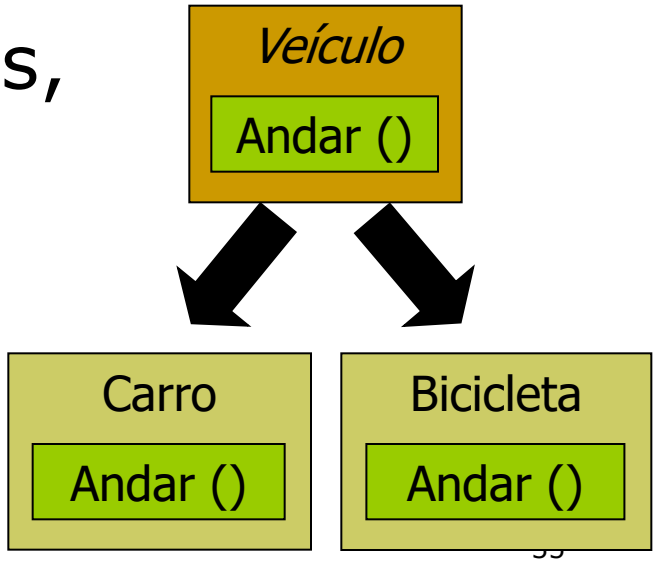
Polimorfismo

Sobrecarga

Sobrescrita

Classes abstratas

- Classes abstratas não podem ser instanciadas;
  - Usadas para organizar características comuns a diversas subclasses;
  - Desenvolvida para ser herdada.
- Não possui instâncias diretas, só indiretas.

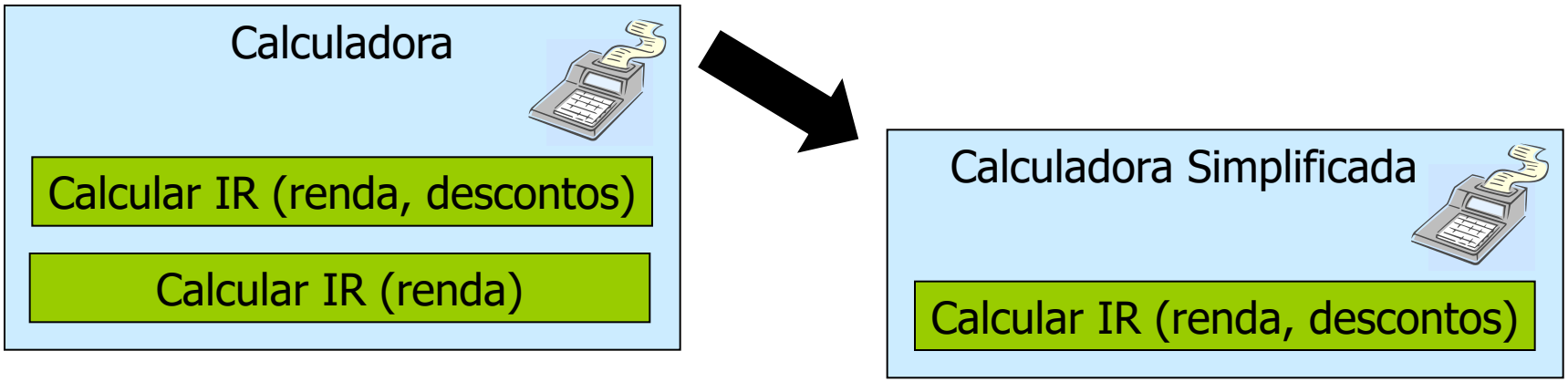


Operações abstratas

- Classes abstratas podem definir métodos sem implementação, chamados abstratos;
  - Subclasses concretas são obrigadas a implementá-lo;
  - Classes concretas não podem ter métodos abstratos;
  - Classes abstratas podem ter métodos concretos.

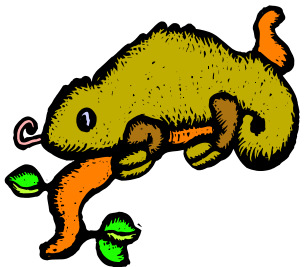
Sobrecarga e sobrescrita

- Sobrecarga: operações distintas com o mesmo nome;
- Sobrescrita: subclasse define nova implementação para operação definida na superclasse.



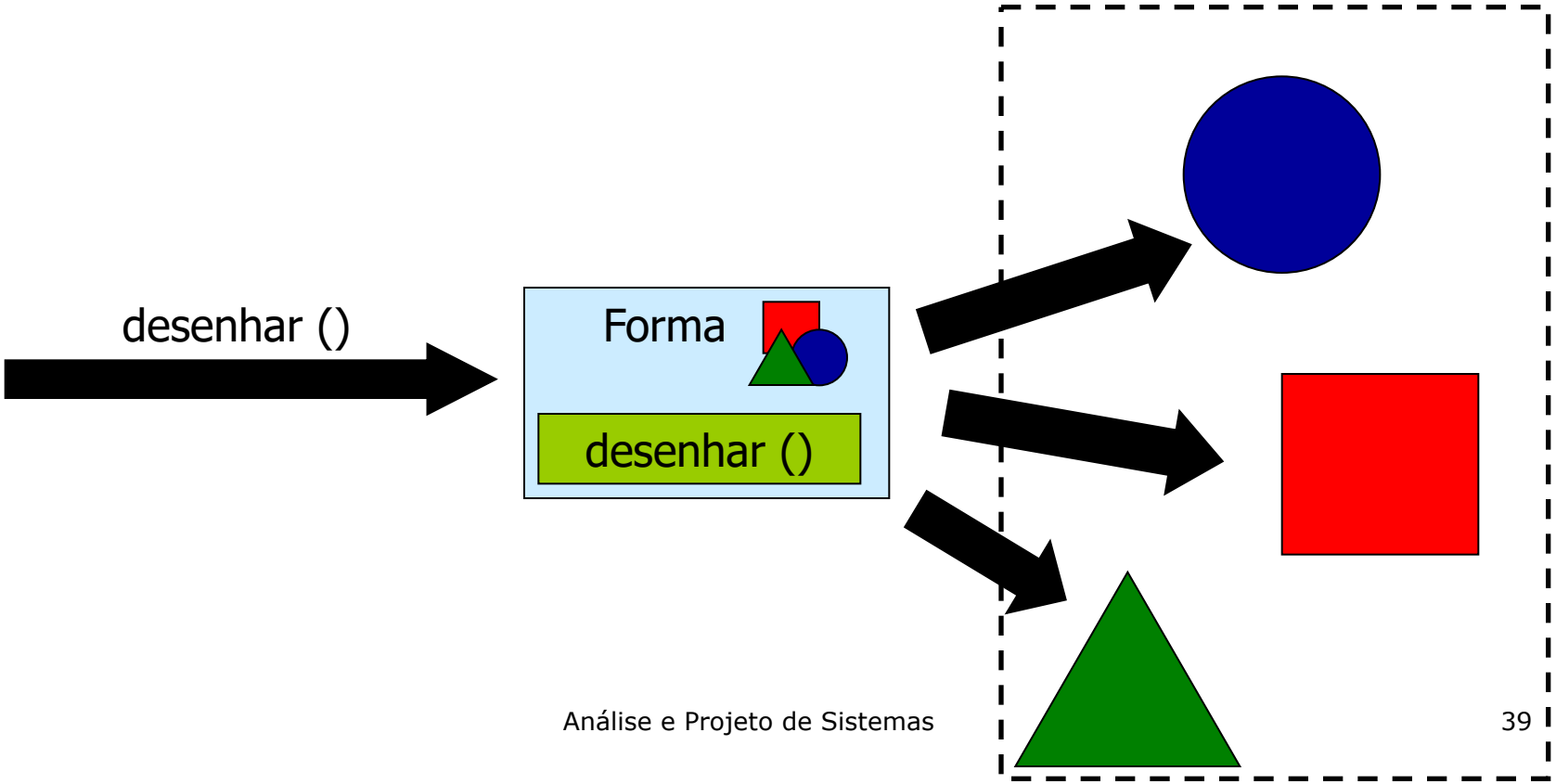
Polimorfismo

- O objeto emissor não precisa saber quem é o objeto receptor, contanto que saiba que ele responde a uma certa mensagem;
  - O emissor só conhece uma interface;
  - O receptor sabe a implementação certa.
- É viabilizado através de sobrecarga e sobrescrita.



Polimorfismo

- Habilidade de tomar várias formas.



Dicionário de dados

- Incluído na especificação de análise;
- Descreve cada atributo:
  - Seu significado no domínio;
  - Unidades de medida;
  - Intervalo / limite;
  - Enumeração;
  - Precisão;
  - Valor default;
  - Obrigatoriedade;
  - Etc.

Separação em subsistemas

- Projetos grandes podem conter centenas de classes e estruturas diversas;
- Divisão das classes em pacotes:
  - Coleção de classes que colaboram entre si;
  - Conjunto coeso de responsabilidades;
  - "Caixa preta".
- Vantagens:
  - Facilita o entendimento para leitores;
  - Auxilia na organização de grupos de trabalho;
  - Organiza a documentação.

Pacotes de classes e de caso de uso

- Casos de uso podem ser um bom ponto de partida para definição do agrupamento;
- No entanto, não é obrigatório que a divisão seja a mesma;
- Apresenta-se um diagrama de pacotes e cada pacote possui um diagrama de classes próprio.