nemo

ontology & conceptual modeling research group

O Paradigma Orientado a Objetos

Vítor E. Silva Souza

(vitorsouza@inf.ufes.br)

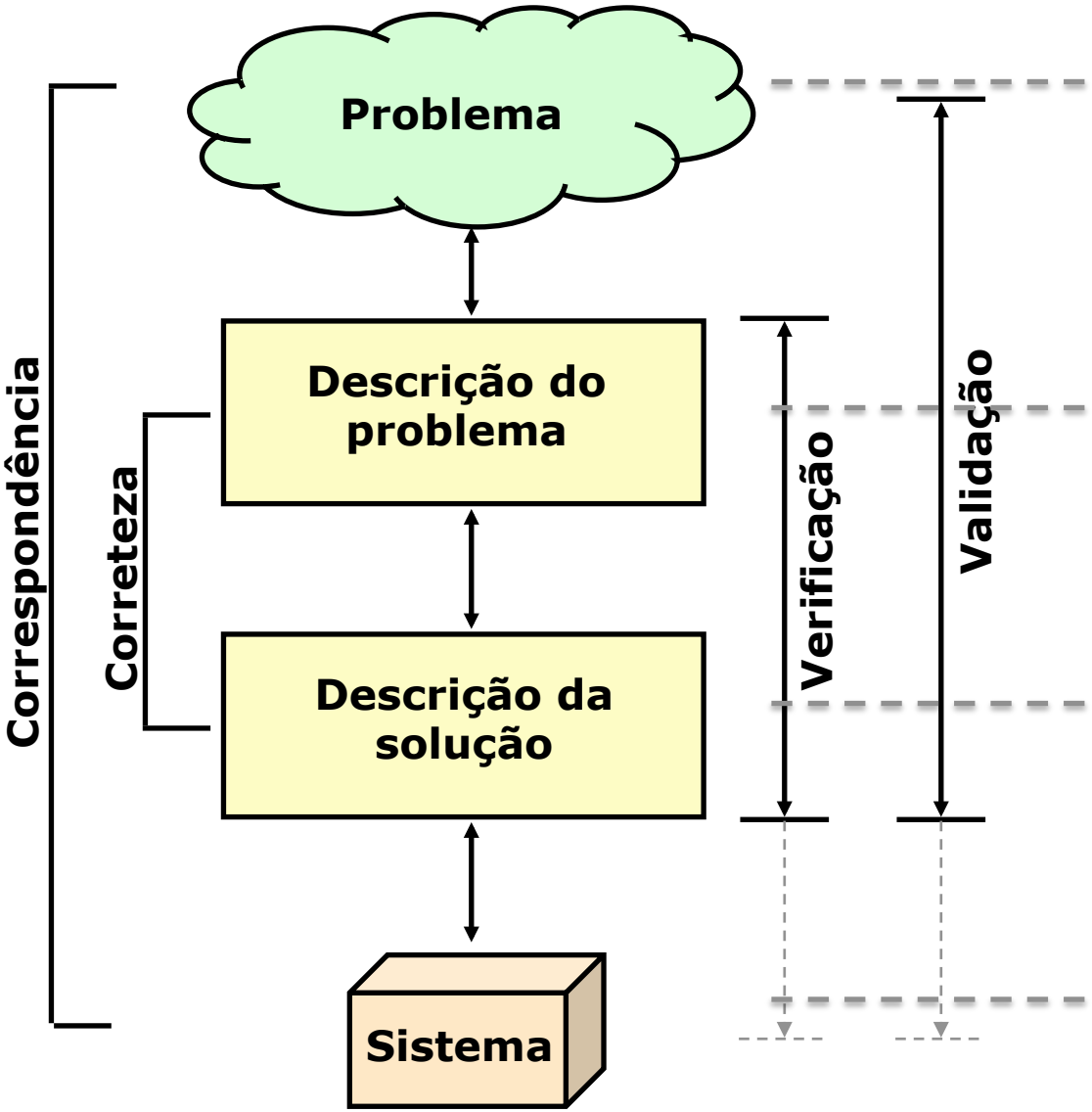
http://www.inf.ufes.br/~vitorsouza

Departamento de Informática

Centro Tecnológico

Universidade Federal do Espírito Santo

Contexto: desenvolvimento de sistemas



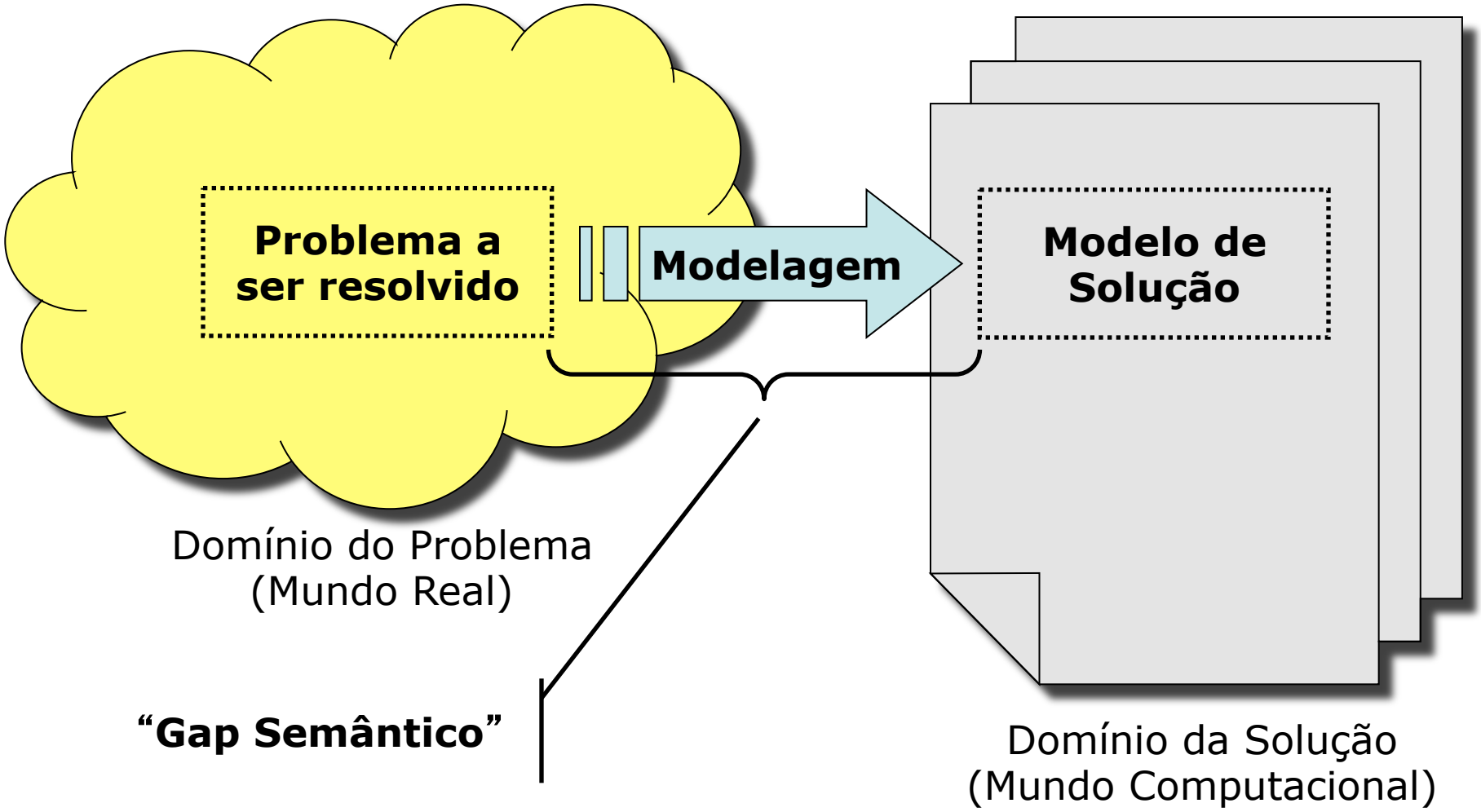
Gestão de serviços de uma biblioteca.

Toda solicitação de empréstimo deve ser atendida.

Processo de check-out de livros da biblioteca.

Sistema de check-out de livros da biblioteca.

Modelagem e o gap semântico



O que é o gap semântico?

- Distância entre o problema no mundo real e o modelo abstrato construído para o mundo computacional;
- Quanto menor, mais rápida será a construção da solução;
- Diminuir o gap semântico é um dos objetivos da Engenharia de Software;
- O paradigma orientado a objetos busca meios de diminuir este gap.

Paradigmas de desenvolvimento

- O que é um paradigma?
  - Um exemplo, um modelo, um padrão;
  - Um conjunto de ideias, uma base filosófica.
- Um paradigma de desenvolvimento agrupa métodos e técnicas que seguem um mesmo conjunto de princípios;
- Dois dos mais conhecidos são:
  - Desenvolvimento Estruturado;
  - Orientação a Objetos (OO).

Paradigma é a representação de um padrão a ser seguido. É um pressuposto filosófico, matriz, ou seja, uma teoria, um conhecimento que origina o estudo de um campo científico; uma realização científica com métodos e valores que são concebidos como modelo; uma referência inicial como base de modelo para estudos e pesquisas. (http://pt.wikipedia.org/wiki/Paradigma)

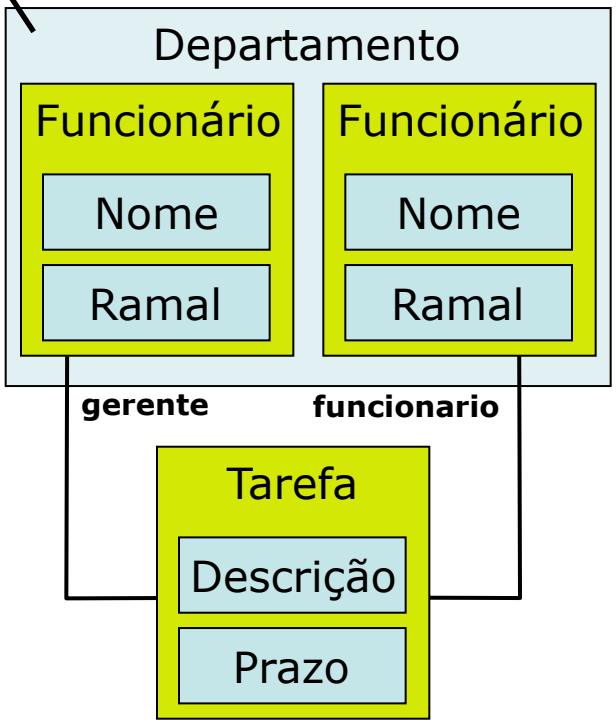
Paradigmas de desenvolvimento

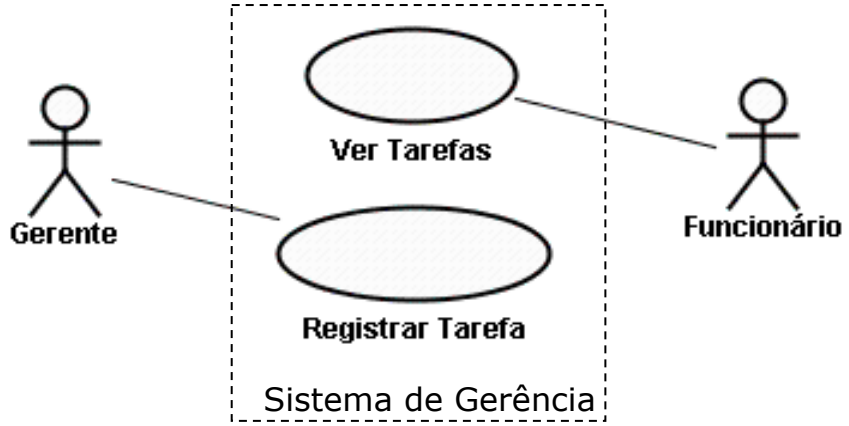
- Estruturado:
  - Modelo entrada – processamento – saída;
  - Dados separados das funções;
  - Visto na disciplina de Programação II / PBC.
- Orientado a Objetos (OO):
  - O mundo é composto por objetos;
  - Objetos combinam dados e funções;
  - Conceitos do problema são modelados como objetos que são associados e interagem entre si.

Paradigma OO vs. estruturado

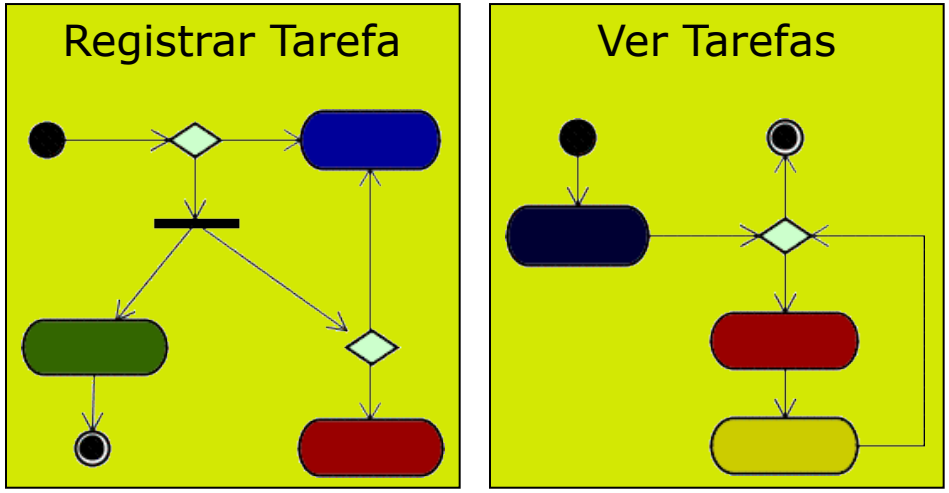
Mais próximo do mundo real. A lógica é encapsulada em objetos.

Orientado a Objetos





Estruturado / Procedural



Desvantagens do paradigma estruturado

- O gap semântico é maior;
- Frequentemente gera sistemas difíceis de serem mantidos:
  - As funções tem que conhecer a estrutura dos dados;
  - Mudanças na estrutura dos dados acarreta alteração em todas as funções relacionadas.

Por estes motivos, o paradigma orientado a objetos vem tomando o espaço que antes era dominado pelo paradigma estruturado.

Benefícios da orientação a objetos

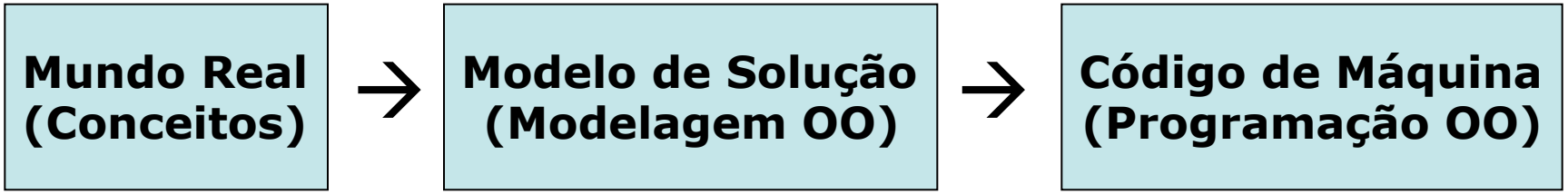
- Capacidade de enfrentar novos domínios;
- Melhoria da interação analistas x especialistas;
- Aumento da consistência interna da análise;
- Alterabilidade, legibilidade e extensibilidade;
- Apoio à reutilização.

Orientação a objetos não é mágica e nem a "tábua de salvação" do desenvolvimento. É preciso aplicá-la com disciplina e em conjunto com outras técnicas da Engenharia de Software.

O foco deste curso é na programação OO. No entanto, é preciso entender os conceitos por trás do paradigma e aprender a modelar soluções OO. Você também usará isso no futuro!

Filosofia

- "O mundo é composto por objetos";



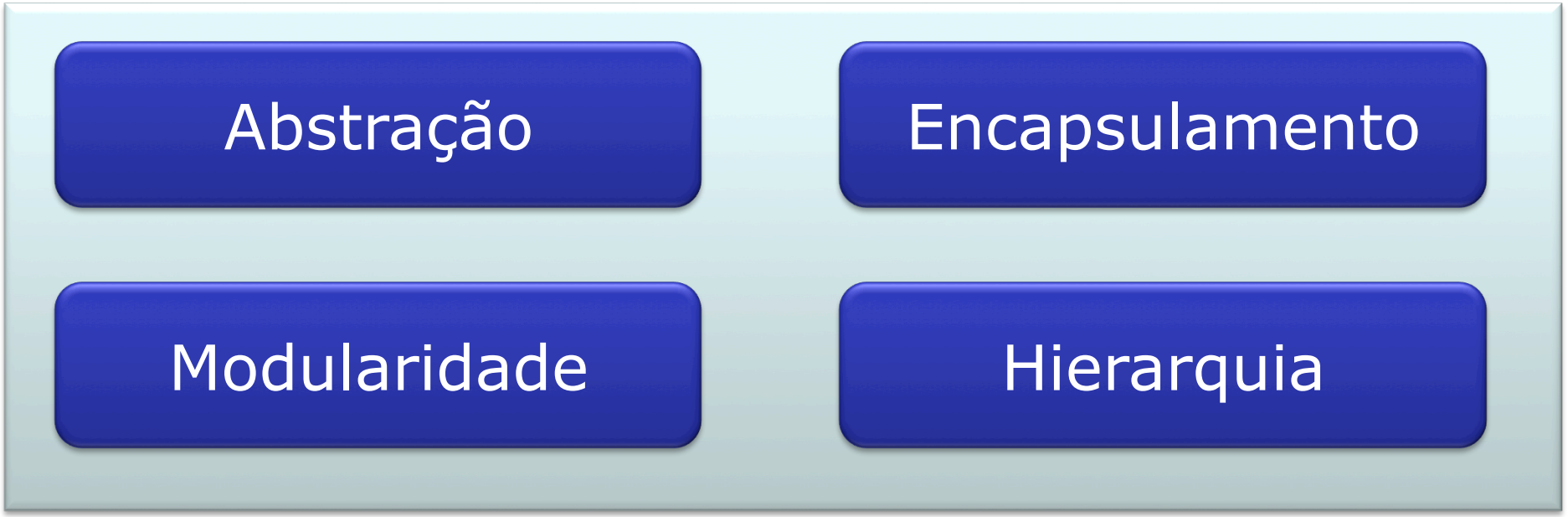
- OO tenta gerenciar a complexidade dos problemas do mundo real abstraindo o conhecimento relevante e encapsulando-o em objetos.

"Um sistema construído usando um método orientado a objetos é aquele cujos componentes são partes encapsuladas de dados e funções, que podem herdar atributos e comportamento de outros componentes da mesma natureza, e cujos componentes comunicam-se entre si por meio de mensagens".

Eduard Yourdon

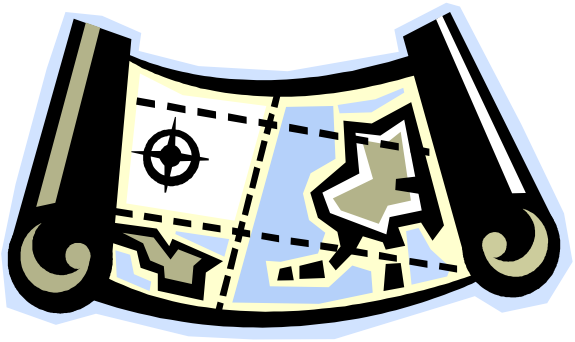
Princípios fundamentais

- Auxiliam a administrar a complexidade;
- Guiam toda a tarefa de modelagem.



Abstração

- "Modelos mentais": visão simplificada do mundo construída por cada um em cada situação;
- Abstrair consiste em ignorar aspectos irrelevantes e concentrar nos principais.

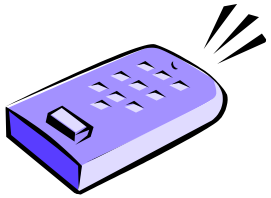


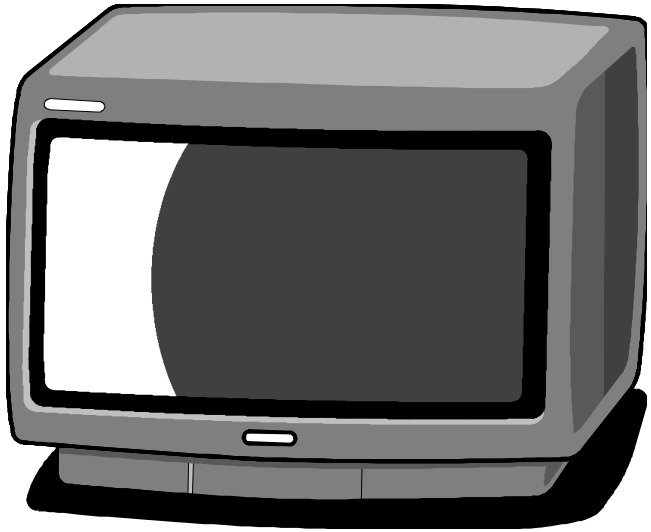
Abstração

- Abstração de Dados:
  - Um tipo é definido por suas operações;
  - Ex.: Um tipo pilha é definido por suas operações `empilhar` e `desempilhar`.
- Abstração de Procedimentos:
  - Uma operação com efeito bem definido pode ser tratada como atômica, mesmo que ela faça uso de outras operações de mais baixo nível;
  - `calcularSalarioLiquido`: definida em termos de `obterSalarioBruto`, `calcularImposto`, `calcularDescontos`, etc.

Encapsulamento

- Separar os aspectos externos (o que faz) dos aspectos internos (como faz):
  - Aspectos externos = interface, contrato;
  - Aspectos internos = implementação.



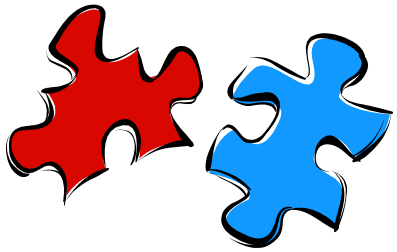


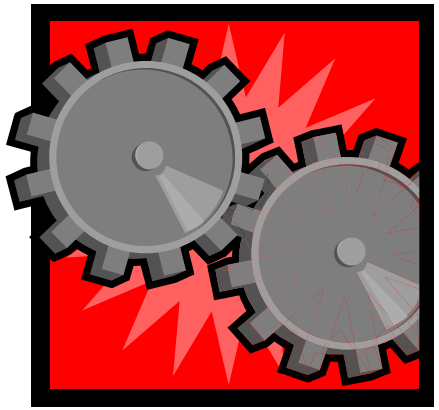
Encapsulamento

- Complemento da abstração:
  - Abstração enfoca o comportamento observável de um objeto;
  - Encapsulamento enfoca a implementação que origina este comportamento.
- Promove maior estabilidade:
  - Clientes do objeto só conhecem sua interface;
  - Podemos alterar a implementação de uma operação sem afetar o restante do sistema.

Modularidade

- Decomposição do sistema em módulos:
  - Coesos (baixo acoplamento);
  - Autônomos;
  - De interface simples e coerente.
- Fundamental para o reuso e extensão.





Hierarquia

- É uma forma de arrumar as abstrações e simplificar o entendimento do problema;
- Também promovem o reuso;
- Sinergia para administrar a complexidade:
  - Abstração auxilia a identificar os conceitos relevantes do mundo real;
  - Encapsulamento oculta a visão interna das abstrações identificadas;
  - Modularidade nos dá um meio de agrupar logicamente abstrações relacionadas;
  - Por fim, abstrações formam hierarquias.

Conceitos básicos

Classes

Instâncias

Objetos

Estruturação

Associação

Composição

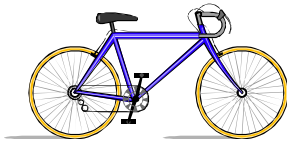
Herança

Mensagens

Métodos

Objetos

- “Um objeto é uma entidade que incorpora uma abstração relevante no contexto de uma aplicação”;
- Podem ser coisas abstratas (ex.: uma reserva de passagem aérea) ou concretas (ex.: um documento).

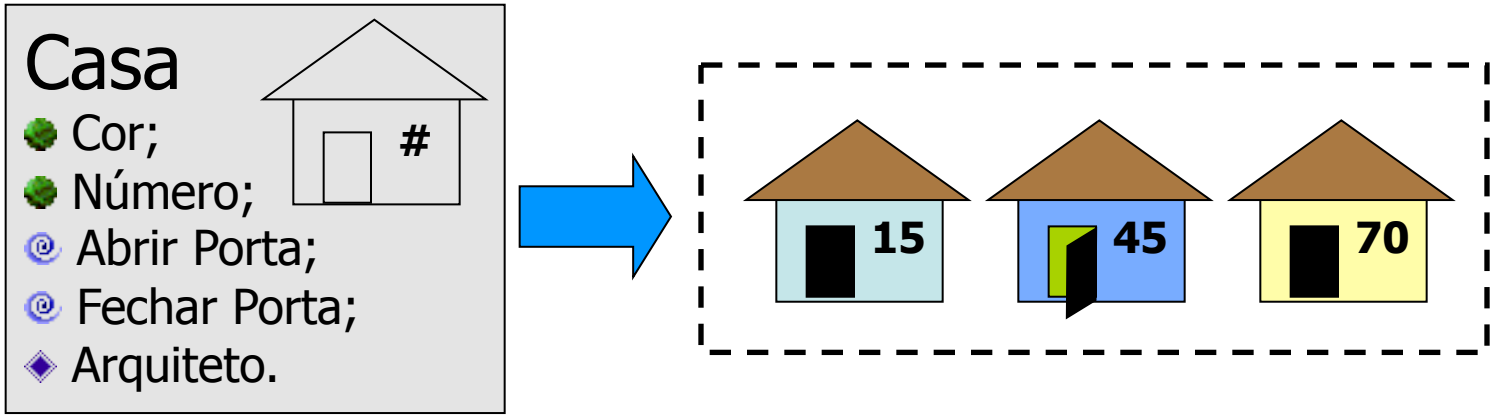


Objetos – características principais

- Estado (estrutura): conjunto de suas propriedades e seus valores correntes;
- Comportamento: conjunto de serviços (operações) que o objeto provê;
- Identidade: identificador único que diferencia cada objeto, mesmo que tenham o mesmo estado e comportamento.

Classes

- Uma classe descreve um conjunto de objetos com as mesmas propriedades, o mesmo comportamento, os mesmos relacionamentos com outros objetos e a mesma semântica;
- Similar ao conceito de tipo.

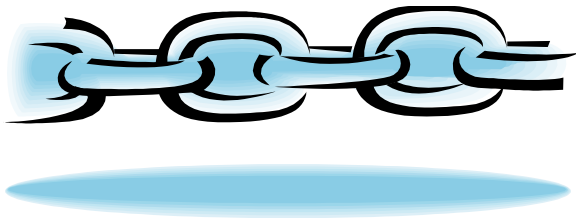


Classes e instâncias

- Objeto = Instância de classe;
- Paradigma OO norteia o desenvolvimento por meio de classificação de objetos:
  - Modelamos classes, e não objetos;
  - Objetos são entidades reais – executam algum papel no sistema;
  - Classes são abstrações – capturam a estrutura e comportamento comum a um conjunto de objetos.

Mecanismos de estruturação

- Objetos relacionam-se uns com os outros;
- É preciso modelar esta complexidade e estruturar as classes;
- Mecanismos propostos:
  - Associação;
  - Composição;
  - Herança.

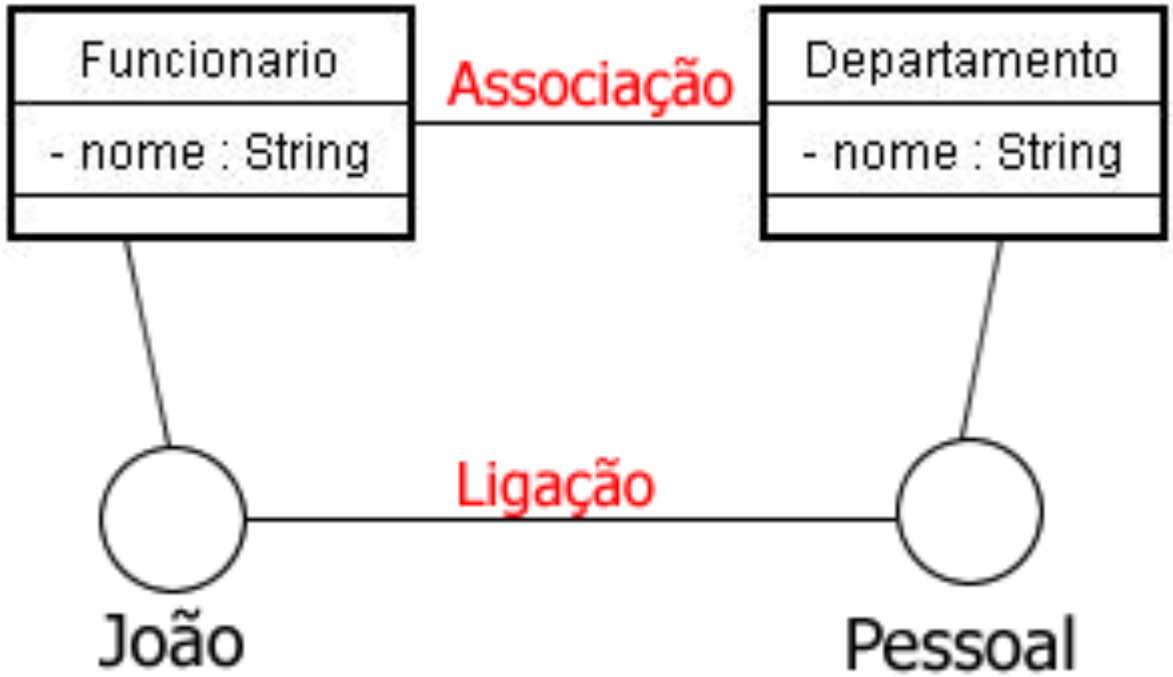


Ligações e associações

- 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].

Ligações e associações

- Em UML...

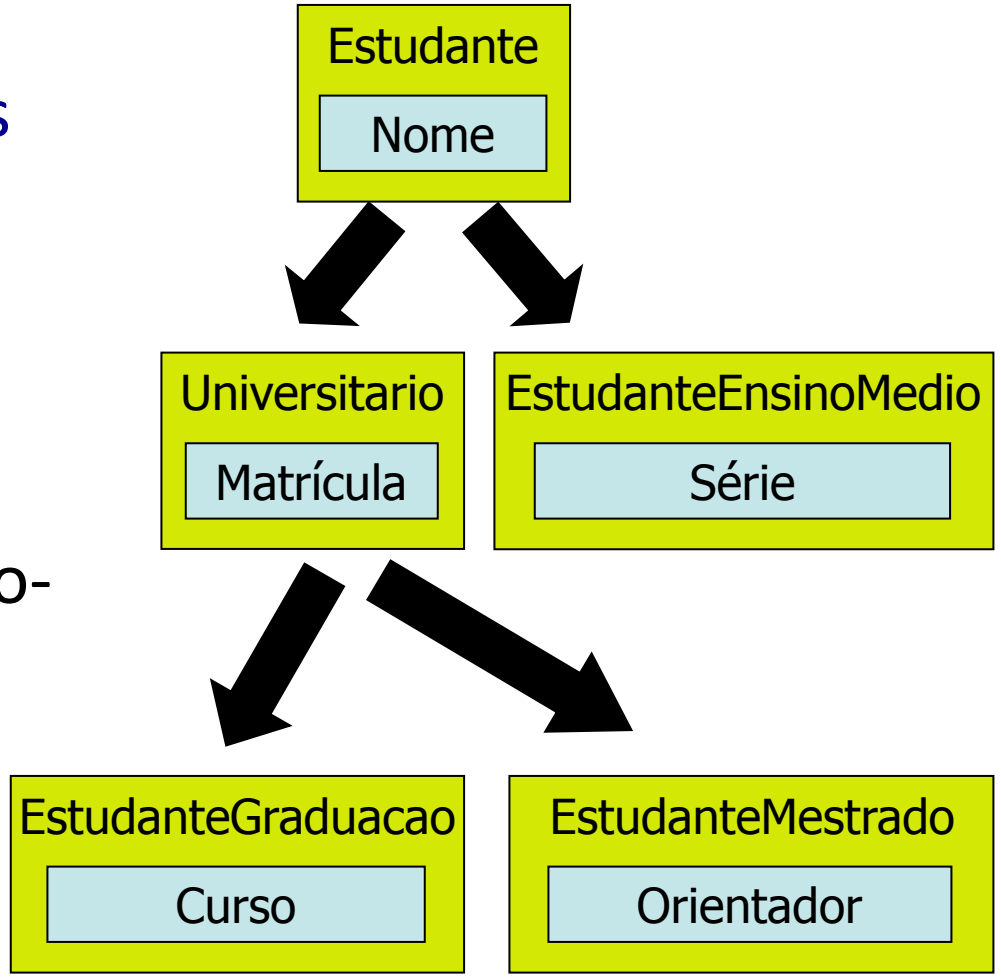


Composição e agregação

- Formas especiais de ligação que modelam relacionamentos todo-parte;
- Objetos complexos agregam ou são compostos de objetos mais simples;
- Composição:
  - A parte só pode pertencer a um todo (por vez). Ex.: uma aula de uma disciplina, um motor de um carro;
  - A parte pode não pertencer a um todo (ex.: motor);
- Agregação:
  - A parte pode ser compartilhada com vários todos. Ex.: uma disciplina de um (ou mais) cursos de grad.

Generalização e especialização

- Generalização: quando classes têm semelhanças podemos definir uma classe mais geral;
- Especialização: muitas vezes um conceito pode ser refinado, adicionando-se novas características.



Herança

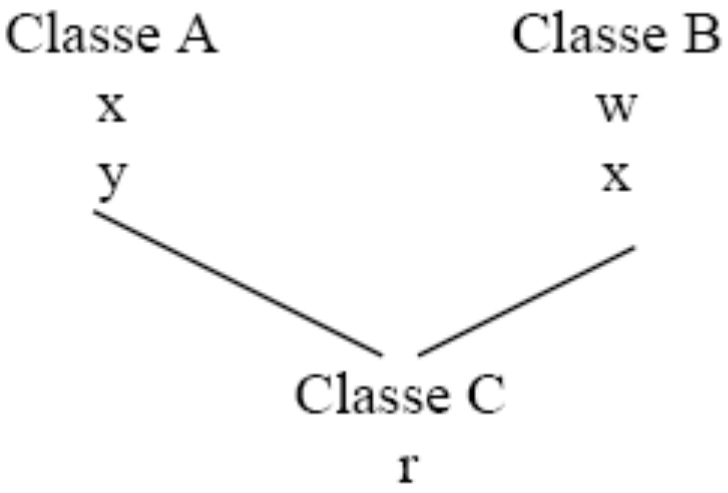
- Generalização e especialização são úteis para estruturação do sistema;
- São formadas hierarquias de classes:
  - "Filhos" (ou subclasses) herdam estrutura e comportamento dos "pais" (ou superclasses) e demais "ancestrais", de forma indireta.
- A herança possibilita:
  - Reutilização;
  - Captura explícita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

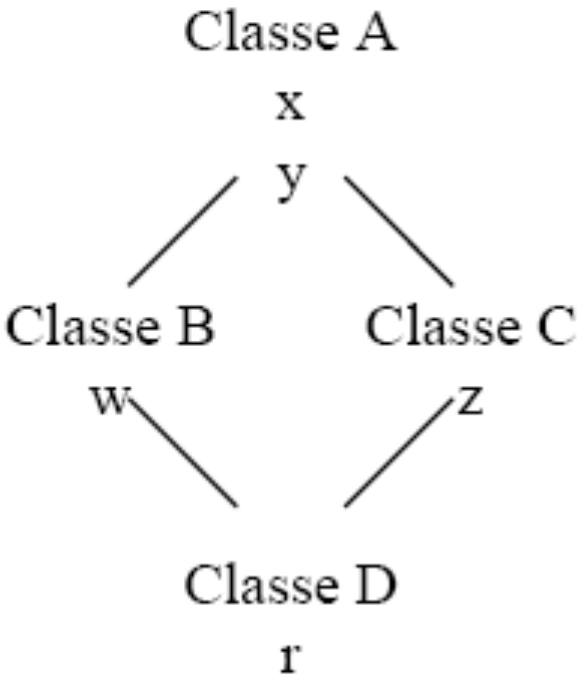
Cuidados com o uso de herança

- Semântica da herança:
  - "é um tipo de";
  - "é uma instância indireta de";
  - Ex.: Universitário é um tipo de Estudante.
- Se a subclasse precisa cancelar características da superclasse, há algo errado;
- Herança pode se tornar um vício!

Herança múltipla

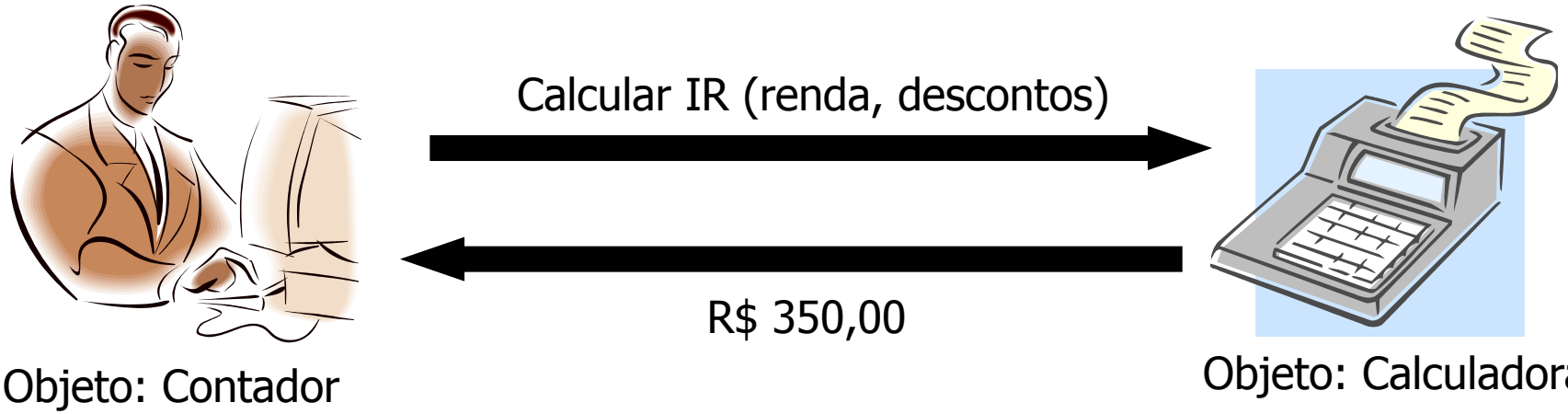
- Atenção à redundância!





Mensagens e métodos

- As operações (serviços) que um objeto oferece são chamadas de métodos;
- Para solicitar um serviço um objeto (cliente) envia uma mensagem a outro.



Mensagens e métodos

- Encapsulamento: não é permitido acessar diretamente as propriedades de um objeto, é preciso operar por meio de métodos (troca de mensagens);
- Abstração: a complexidade de um objeto é escondida "por trás" de suas operações;
- Toda funcionalidade do sistema é realizada pela troca de mensagem entre objetos.

Conceitos avançados

Classes Abstratas

Operações Abstratas

Polimorfismo

Ligação Dinâmica

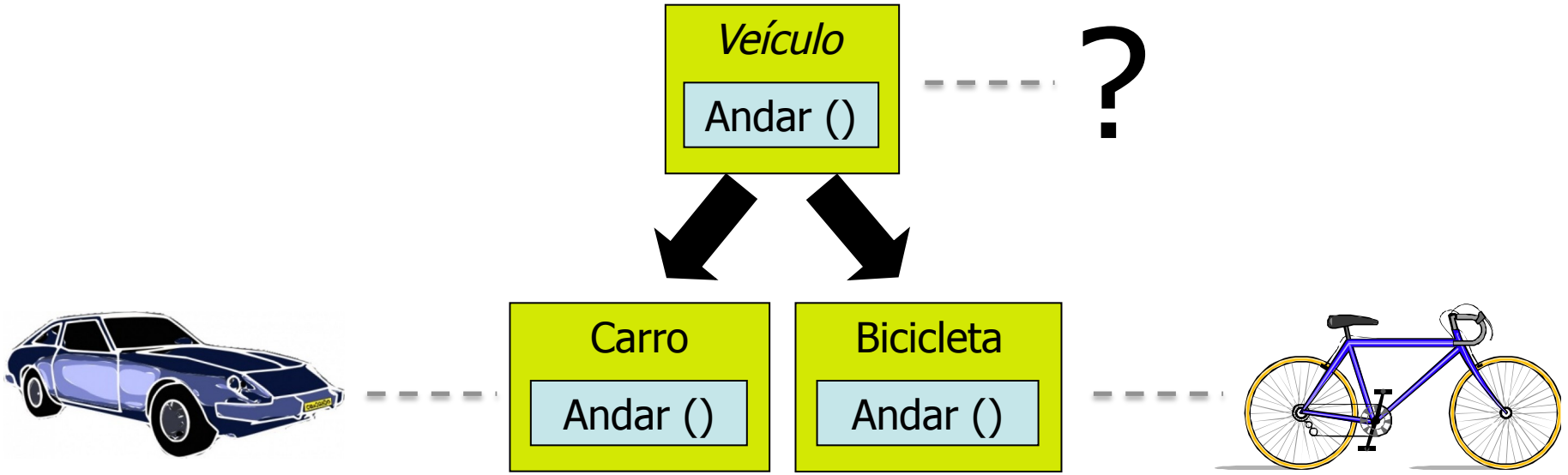
Sobrecarga

Sobrescrita

Persistência

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.

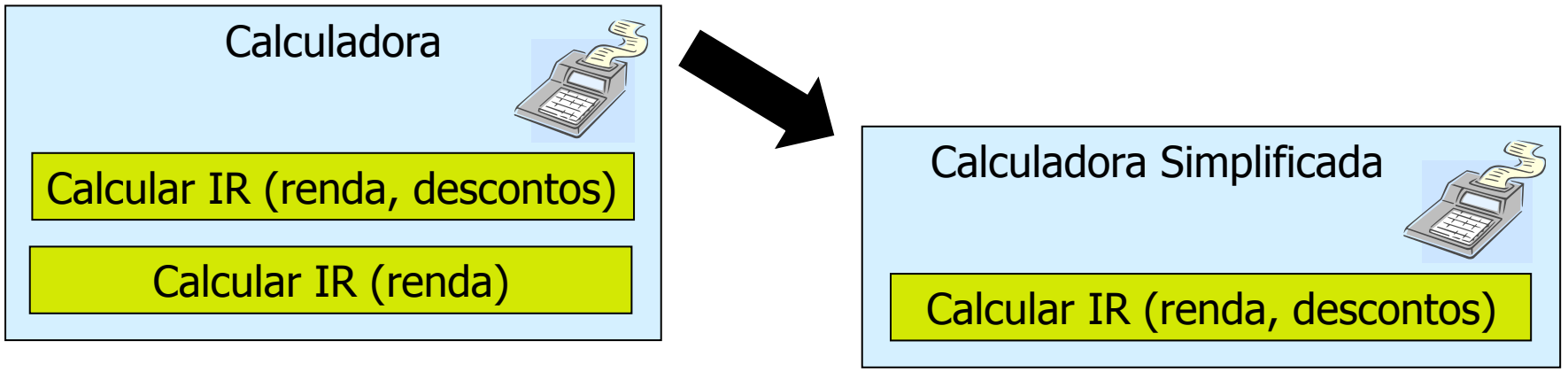


Operações abstratas

- 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.
- Interface = classe abstrata que só possui operações abstratas.

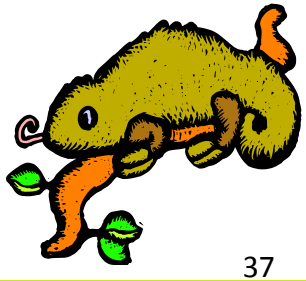
Sobrecarga e sobrescrita

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



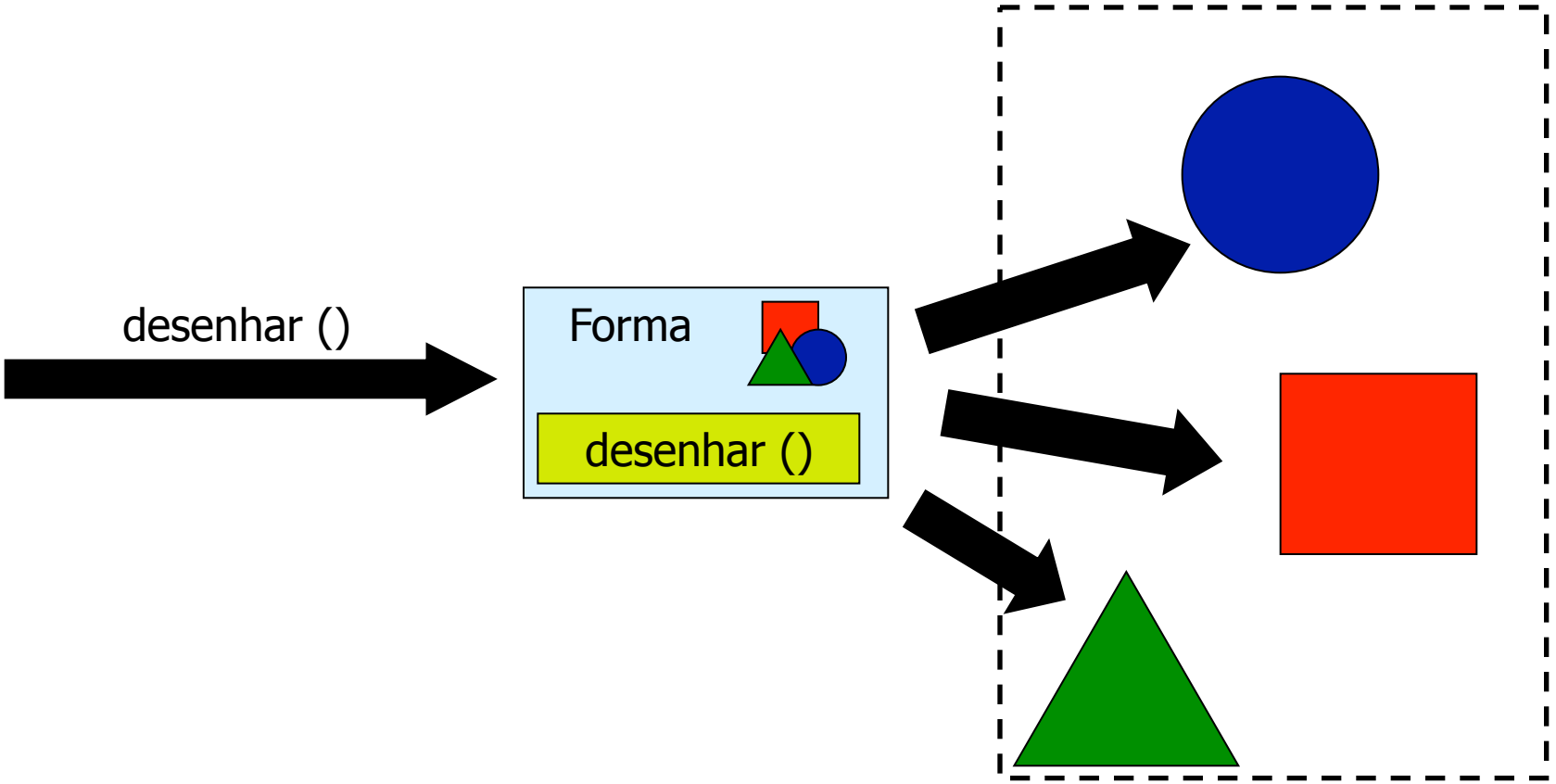
Polimorfismo

- 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 uma forma de sobrescrita, mas todas as operações mantém a mesma semântica em toda a hierarquia.



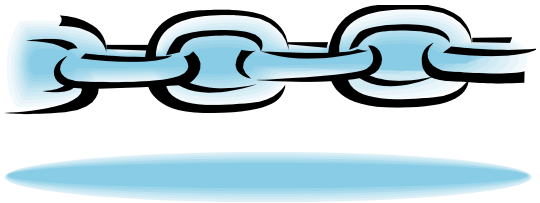
Polimorfismo

- Habilidade de tomar várias formas.



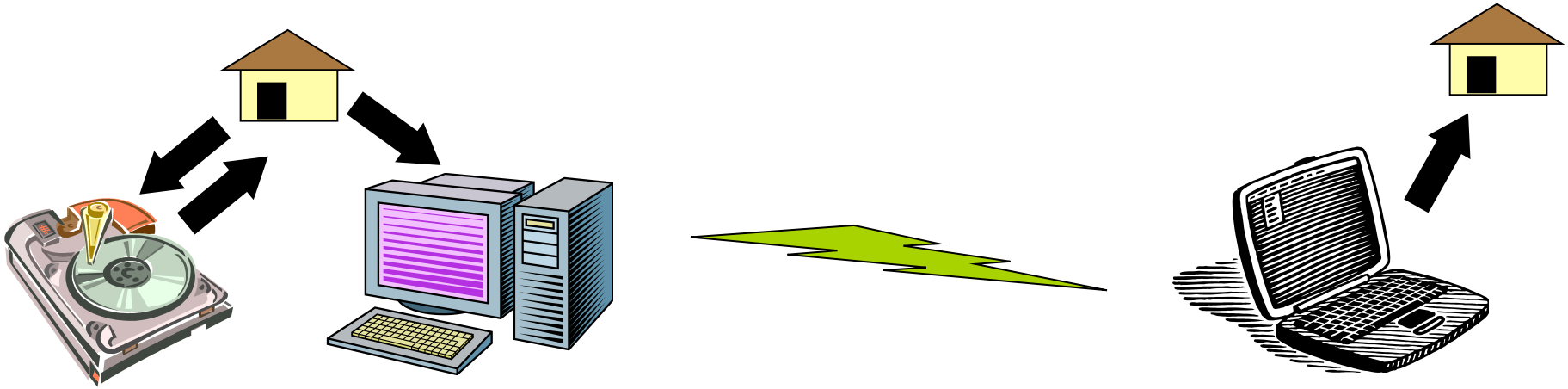
Ligação dinâmica

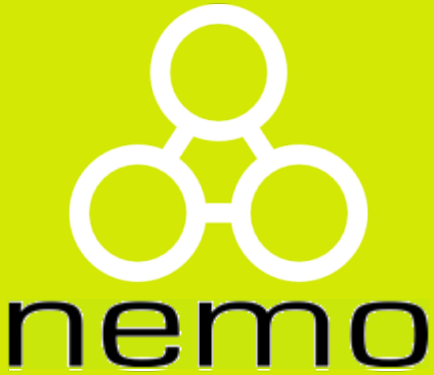
- Quando chamamos uma função, esta deve estar ligada ao código que a implementa;
  - Ligação estática = feita em tempo de compilação;
  - Ligação dinâmica ou tardia = feita em tempo de execução.
- Necessário por nem sempre sabermos a classe verdadeira de um objeto;
- É a base para o polimorfismo.



Persistência

- Capacidade do objeto de transcender o tempo e o espaço;
  - Armazenamento em banco de dados;
  - Transmissão pela rede.





http://nemo.inf.ufes.br/