Introdução a Orientação a Objetos

Reginaldo Ré
reginaldo@utfpr.edu.br

Universidade Tecnológica Federal do Paraná

Análise e Projeto Orientados a Objetos
2018/2

Agenda

1. Introdução
2. Histórico da Evolução da Orientação a Objetos
3. Introdução a Linguagem de Modelagem Unificada – UML

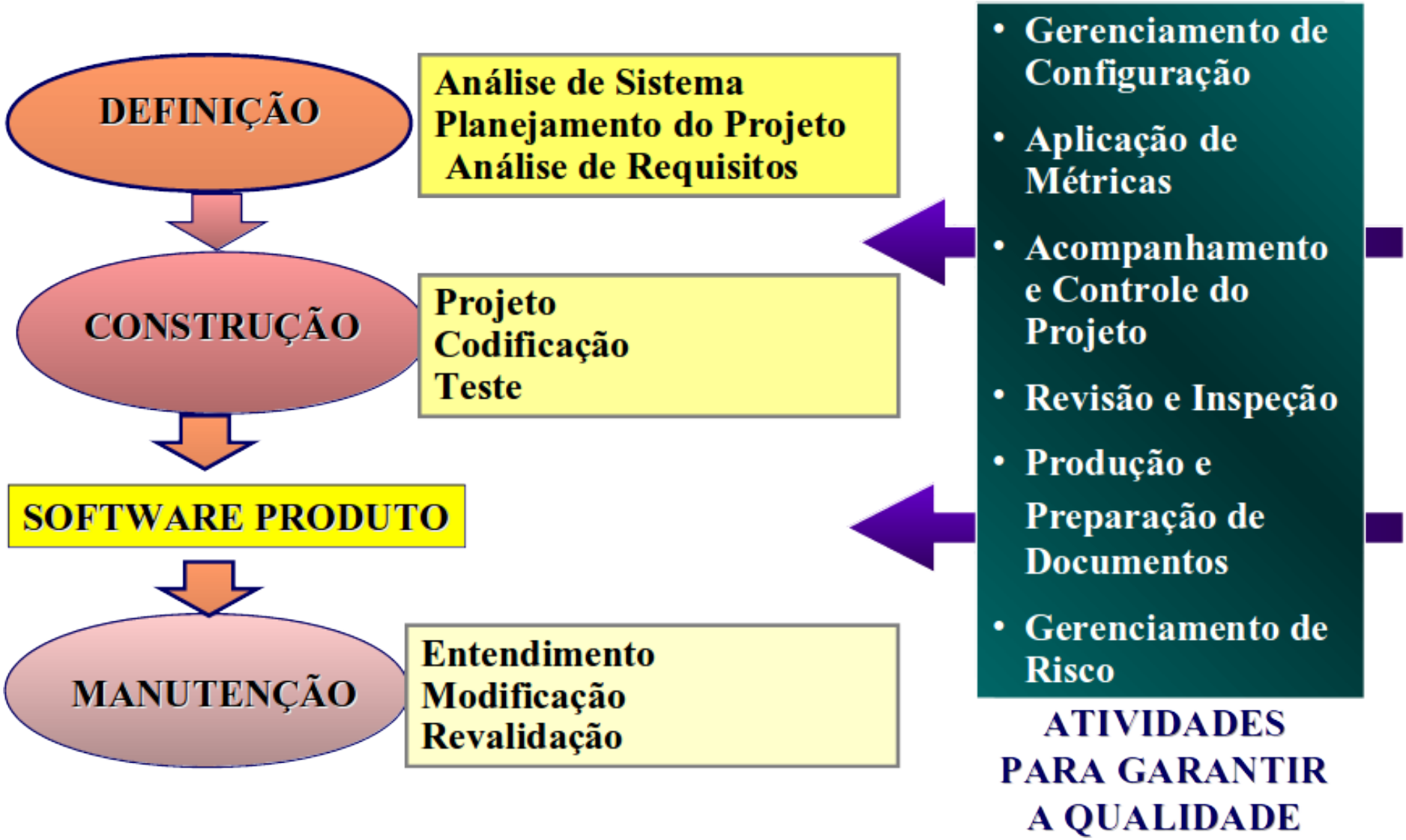
Agenda

Introdução

- Independentemente da natureza do projeto e aplicação os modelos de processo de software possuem:
  - fase de Definição;
  - fase de Construção;
  - fase de Manutenção;
  - atividades de Apoio.

Introdução

Fases dos Modelos de Processo de Software



Introdução

- Necessidade de abordagens para desenvolver software de maneira organizada e estruturada:
  - Análise Estruturada
  - Análise Essencial
  - Análise e Projeto OO

Introdução

Perguntas

- O que é uma abordagem de desenvolvimento de software?
- As abordagens param por aqui?
- Porque estudaremos a abordagem OO?

Agenda

Histórico de OO

- Final da década de 60: Dois cientistas dinamarqueses criaram a linguagem Simula (Simulation Language)
- 1967: Linguagem de Programação Simula-67 – conceitos de classe e herança
- 1983: Termo Programação Orientada a Objeto (POO) é introduzido com a linguagem Smalltalk
- Fim dos anos 80: Paradigma de OO – abordagem poderosa e prática para o desenvolvimento de software
- Linguagens OO
  - Smalltalk (1972), Ada (1983), Eiffel (1985)
  - Object Pascal (1986), Common Lisp (1986), C++ (1986)
  - Java (1990), Phyton (1990), Perl 5 (2005)

- Puras: tudo nelas é tratado consistentemente como um objeto, desde as primitivas até caracteres e pontuação. Exemplos: Smalltalk, Eiffel, Ruby
- Projetadas para OO, mas com alguns elementos procedimentais. Exemplos: Java, Python
- Linguagens historicamente procedimentais, mas que foram estendidas com características OO. Exemplos: C++, Fortran 2003, Perl 5

Histórico OO

- Surgiram vários métodos de análise e projeto OO:
  - CRC (Class Responsability Collaborator, Beck e Cunningham, 1989)
  - OOA (Object Oriented Analysis, Coad e Yourdon, 1990)
  - Booch (1991)
  - OMT (Object Modeling Technique, Rumbaugh, 1991)
  - Objectory (Jacobson, 1992)
  - Fusion (Coleman, 1994)
  - UML + processos como o RUP, UP, XP

Histórico OO

Pergunta

Porque estudar UML?

Agenda

Introdução a UML

- Unified Modeling Language
- Linguagem para especificação, construção, visualização e documentação de sistemas.
- Evolução das linguagens para especificação de conceitos de Booch, OMT e OOSE e também de outros métodos de especificação de requisitos de software OO ou não.

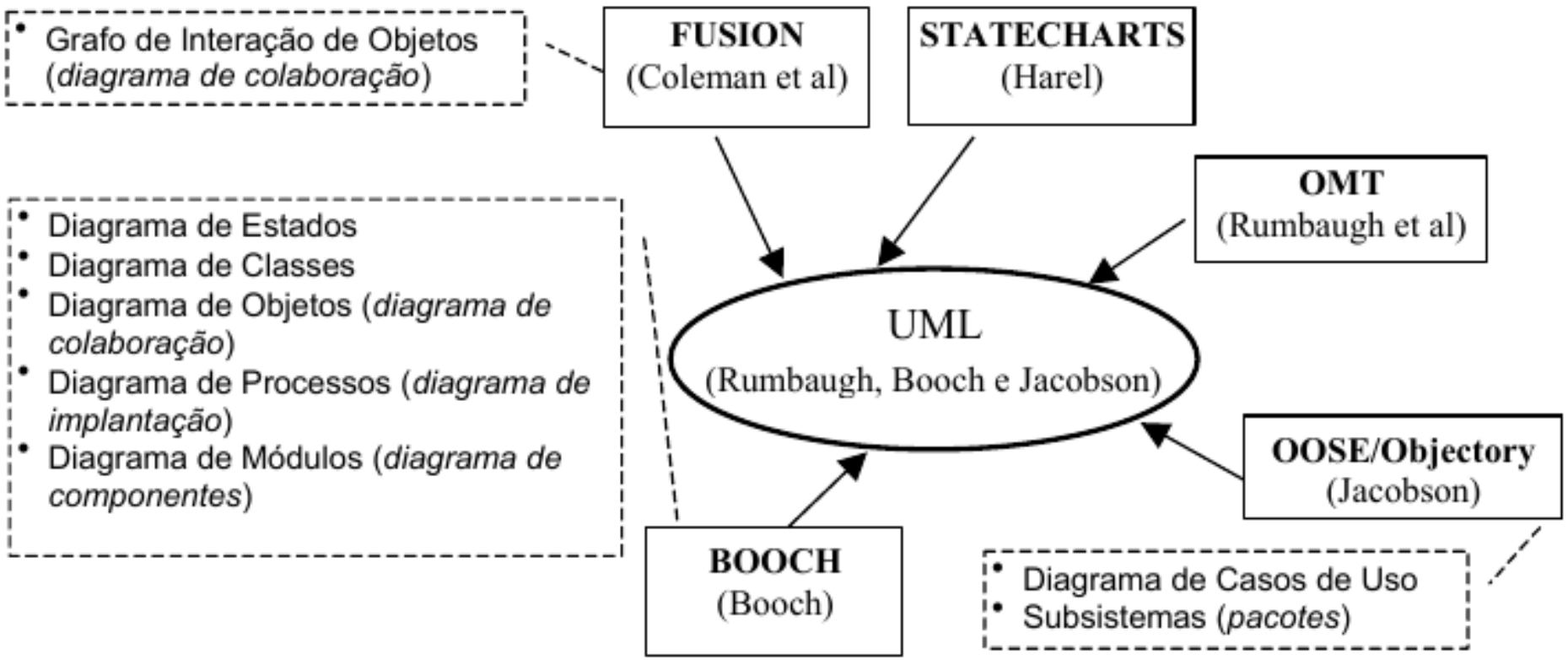
Introdução a UML

Histórico

- Outubro/1994: Booch e Jim Rumbaugh começaram um esforço para unificar o método de Booch e OMT (Object Modeling Language).
- Primeira versão, chamada Unified Method, foi divulgada em outubro/1995.
- Jacobson juntou-se ao grupo, agregando o método OOSE (Object-Oriented Software Engineering).
- Esforço dos três resultou na liberação da UML v. 0.9 e 0.91 em junho e outubro/1996. Em janeiro/1997, foi liberada a versão 1.0 da UML.
- Adotada como padrão segundo a OMG (Object Management Group, `http://www.omg.org/`) em Novembro/1997
- UML 2.0 em 2004 e UML 2.5 (versão atual) em 2015

Introdução a UML

Influências



Ferramentas de Apoio

Diversas empresas lançaram ferramentas para auxiliar a modelagem e projeto de sistemas utilizando UML, gerar código a partir da modelagem e projeto e realizar engenharia reversa, ou seja, obter o modelo em UML a partir do código.

Introdução a UML

Exemplos de Ferramentas de Apoio

- Família Rational Rose gera código em Smalltalk, PowerBuilder, C++, J++ e VB: `http://www-03.ibm.com/software/products/en/rosemod`
- ArgoUML: `http://argouml.tigris.org/`
- Lista de ferramentas que envolvem a UML), entre elas Jude (agora Astah) e Visual Paradigm: `http://www.objectsbydesign.com/toolsumltools_byCompany.html`
- StarUML: `http://staruml.sourceforge.net`

Diagramas Mais Usados da UML

- Diagramas de Casos de Uso
- Diagramas de Classe
- Diagramas de Comportamento
  - Diagrama de Estado
  - Diagrama de Atividade
  - Diagrama de Sequência
  - Diagrama de Comunicação
- Diagramas de Implementação
  - Diagrama de Componente
  - Diagrama de Implantação (Deployment)

Diagramas Mais Usados da UML

Evolução

UML 1.X	**UML 2.x**
Atividades	Atividades
Caso de Uso	Caso de Uso
Classe	Classe
Objetos	Objetos
Sequência	Sequência
Colaboração	Comunicação
Transição de Estados	Transição de Estados
	Pacotes
Componentes	Componentes
Implantação	Implantação
	Interatividade
	Tempo

Conceitos Básicos de Orientação a Objetos

Orientação a Objetos

É uma abordagem de programação que procura explorar nosso lado intuitivo.

- Objeto no mundo físico é tipicamente um produtor e consumidor de itens de informação
- Objeto no mundo do software:
  - Qualquer coisa, real ou abstrata, a respeito da qual armazenamos dados e métodos que os manipulam (Martin e Odell, 1995)
- Objetos da computação são análogos aos objetos existentes no mundo real

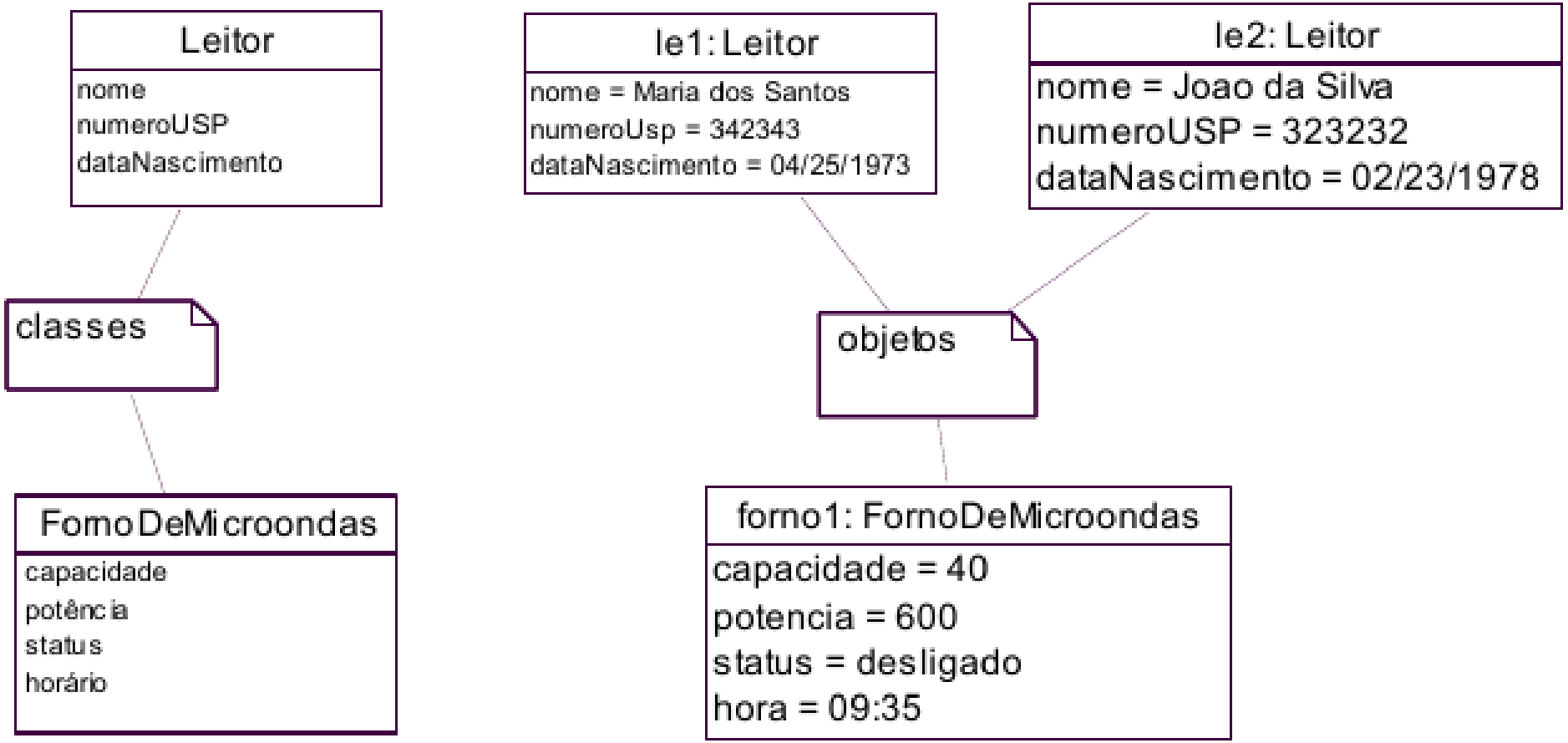
Conceitos Básicos de Orientação a Objetos

Classes

- Agrupamento de objetos similares
- Todo objeto é uma instância de uma Classe
- Objetos representados por determinada classe diferenciam-se entre si pelos valores de seus atributos
- Conjunto de objetos que possuem propriedades semelhantes (ATRIBUTOS), o mesmo comportamento (MÉTODOS), os mesmos relacionamentos com outros objetos e a mesma semântica.

Conceitos Básicos de Orientação a Objetos

Objetos e Classes



Conceitos Básicos de Orientação a Objetos

Atributos

- Representam um conjunto de informações, ou seja, elementos de dados que caracterizam um objeto
- Descrevem as informações que ficam escondidas em um objeto para serem exclusivamente manipuladas pelas operações daquele objeto
- São variáveis que definem o estado de um objeto, ou seja, são entidades que caracterizam os objetos
- Cada objeto possui seu próprio conjunto de atributos

Conceitos Básicos de Orientação a Objetos

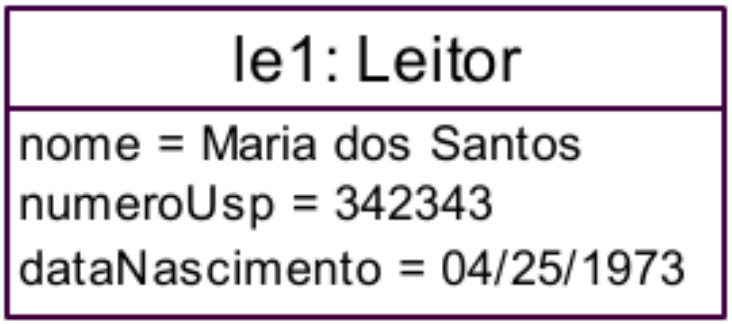
Métodos

- São procedimentos definidos e declarados que atuam sobre um objeto ou sobre uma classe de objetos
- Métodos são invocados por Mensagens
- Cada objeto possui seu próprio conjunto de métodos

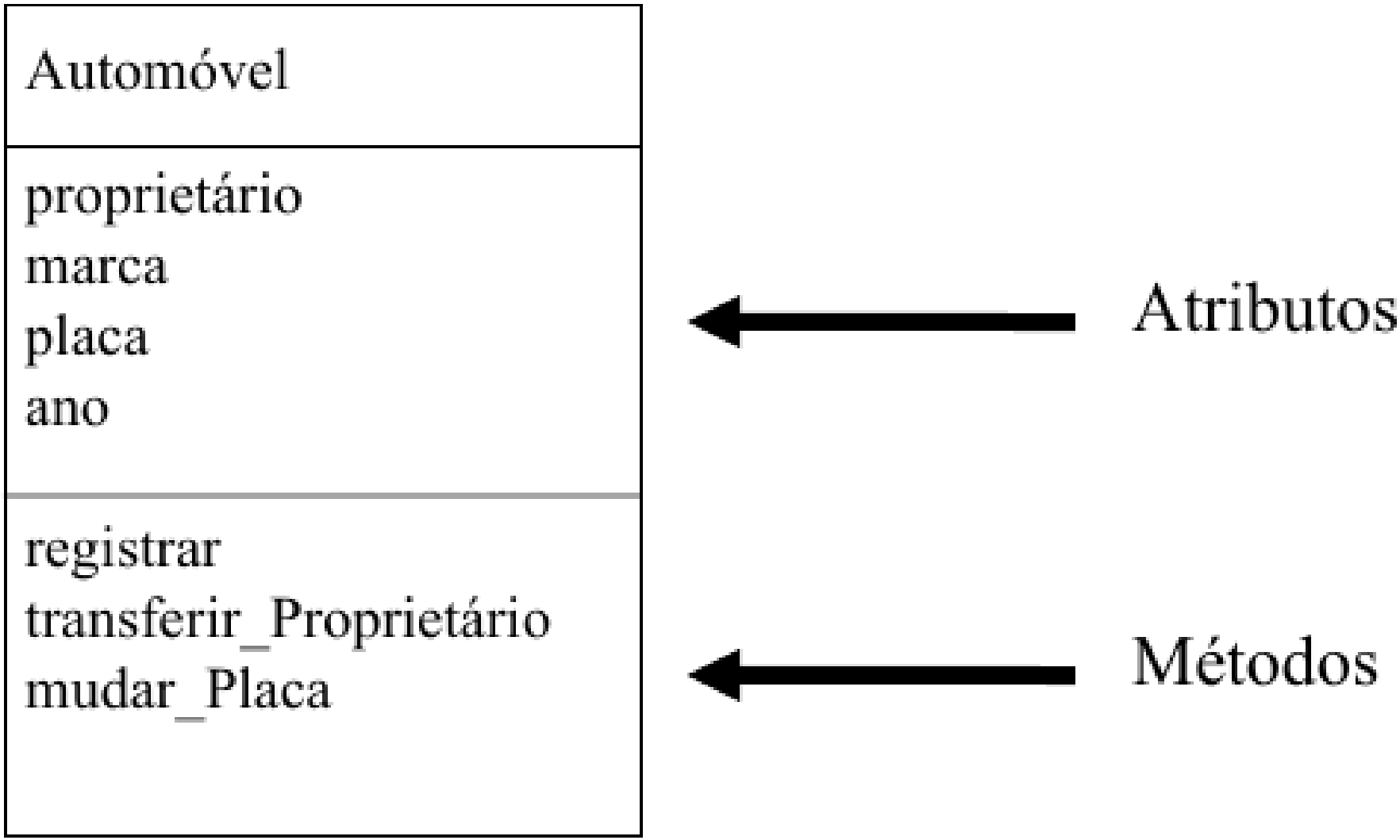
Conceitos Básicos de Orientação a Objetos

Métodos X Mensagem


```
le1.alterarNome('Rosa Olivera')
```




```
método alterarNome(Char[30] novoNome)
Início
    nome := novoNome;
Fim
```


Automóvel
proprietário
marca
placa
ano
registrar
transferir_Proprietário
mudar_Placa
Atributos
Métodos

Conceitos Básicos de Orientação a Objetos

Abstração

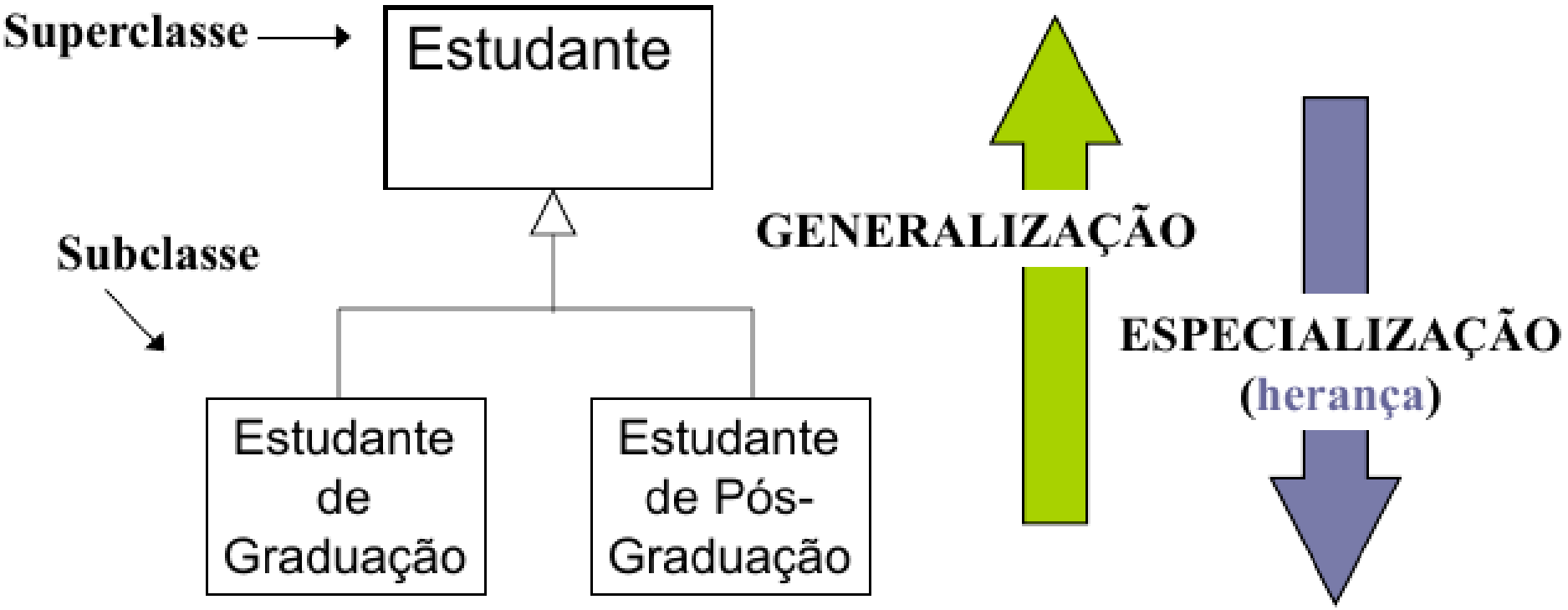
- Processo pelo qual conceitos gerais são formulados a partir de conceitos específicos
- Detalhes são ignorados, para nos concentrarmos nas características essenciais dos objetos de uma coleção

- Permite que certas características ou propriedades dos objetos de uma classe não possam ser vistas ou modificadas externamente, ou seja, ocultam-se as características internas do objeto
  - outras classes só podem acessar os atributos de uma classe invocando os métodos públicos
  - restringe a visibilidade do objeto, mas facilita o reúso

- Mecanismo que permite que características comuns a diversas classes sejam colocadas em uma classe base, ou superclasse.
- As propriedades da superclasse não precisam ser repetidas em cada subclasse.
- Por exemplo, JanelaRolante e JanelaFixa são subclasses de Janela. Elas herdam as propriedades de Janela, como uma região visível na tela. JanelaRolante acrescenta uma barra de paginação e um afastamento.

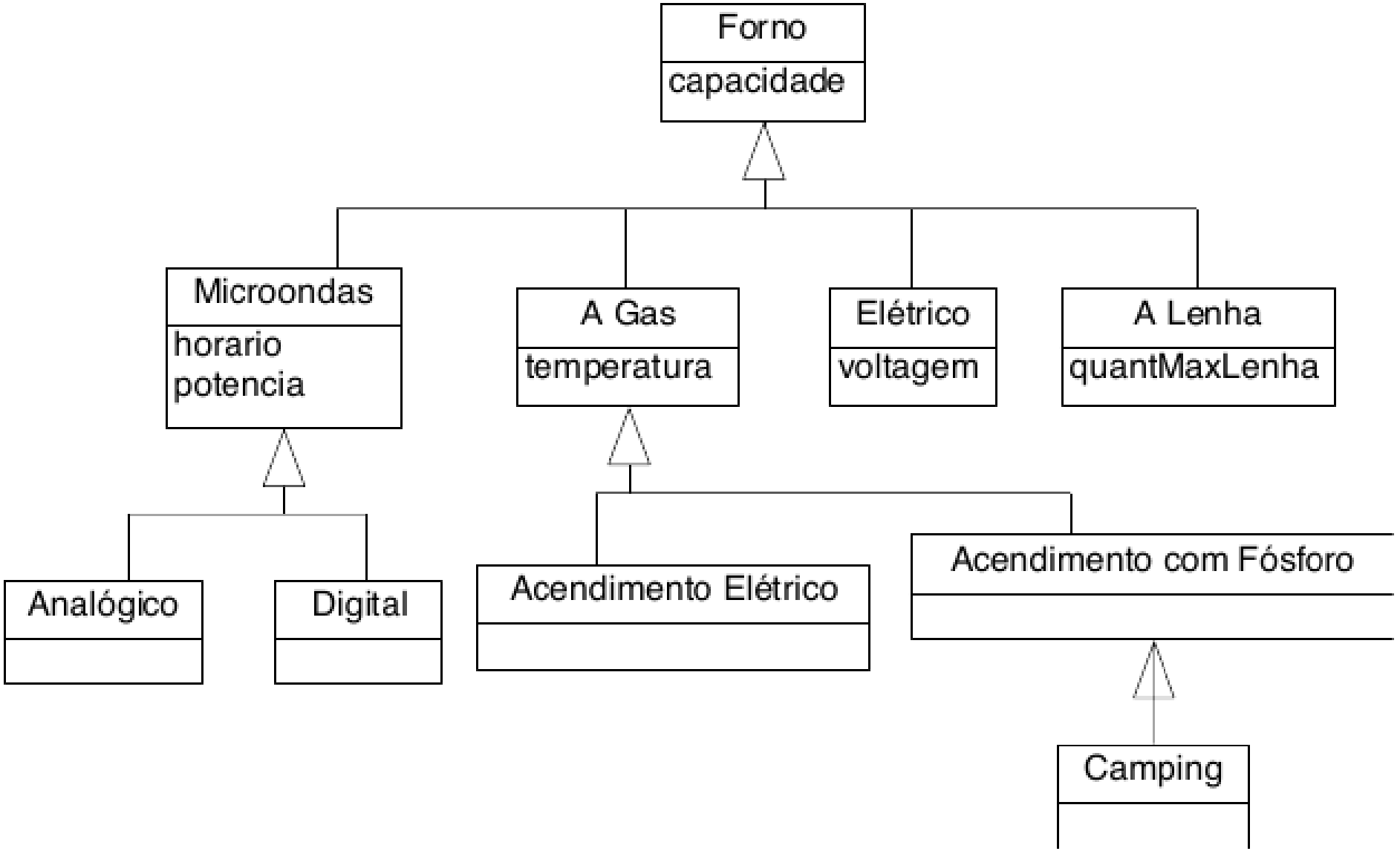
Conceitos Básicos de Orientação a Objetos

Herança: Generalização X Especialização



Conceitos Básicos de Orientação a Objetos

Herança



Vantagens do uso de OO I

- abstração de dados: os detalhes referentes às representações das classes serão visíveis apenas a seus atributos;
- compatibilidade: as heurísticas para a construção das classes e suas interfaces levam a componentes de software que são fáceis de combinar;
- diminuição da complexidade: as classes delimitam-se em unidades naturais para a alocação de tarefas de desenvolvimento de software;
- reutilização: o encapsulamento dos métodos e representação dos dados para a construção de classes facilitam o desenvolvimento de software reutilizável, auxiliando na produtividade de sistemas;

Vantagens do uso de OO II

- extensibilidade: facilidade de estender o software devido a duas razões:
  - herança: novas classes são construídas a partir das que já existem;
  - as classes formam uma estrutura fracamente acoplada, o que facilita alterações;
- manutenibilidade: a modularização natural em classes facilita a realização de alterações no software.
- maior dedicação à fase de análise, preocupando-se com a essência do sistema;
- mesma notação é utilizada desde a fase de análise até a implementação.