



Departamento de Sistemas de Computação
Universidade de São Paulo
Análise e Projeto Orientados a Objetos

Aula 1

Introdução à Orientação a Objetos (OO) e UML

Responsável

Prof. Seiji Isotani (sisotani@icmc.usp.br)

Introdução

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

Fases dos Modelos de Processo de Software

DEFINIÇÃO

Análise de Sistema

Planejamento

Análise de Requisitos

Análise OO

CONSTRUÇÃO

Projeto
Codificação
Teste

Projeto OO

MANUTENÇÃO

Entendimento
Modificação
Revalidação

ATIVIDADES DE APOIO

- *Controle e Acompanhamento do Projeto de Software*
- *Revisões Técnicas Formais*
- *Garantia de Qualidade de Software*
- *Gerenciamento de Configuração de Software*
- *Preparação e Produção de Documentos*
- *Gerenciamento de Reusabilidade*
- *Medidas*
- *Gerenciamento de Riscos*

Conceitos Básicos

- Orientação a Objetos (OO) é uma abordagem de programação que procura explorar nosso lado intuitivo. Os objetos da computação são análogos aos objetos existentes no mundo real.
- No enfoque de OO, os átomos do processo de computação são os objetos que trocam mensagens entre si.
- Essas mensagens resultam na ativação de métodos, os quais realizam as ações necessárias.

Conceitos Básicos

- Os objetos que compartilham uma mesma interface, ou seja, respondem as mesmas mensagens, são agrupados em classes.
- Objeto é algo DINÂMICO: é criado por alguém, tem uma vida, e morre ou é morto por alguém. Assim, durante a execução do sistema, os objetos podem:
 - ser construídos
 - executar ações
 - ser destruídos
 - tornar-se inacessíveis

Histórico de Orientação a Objetos (OO)

- A OO surgiu no final da década de **60**, quando dois cientistas dinamarqueses criaram a linguagem Simula (*Simulation Language*)
- **1967** - Linguagem de Programação Simula-67-
conceitos de classe e herança
- O termo Programação Orientada a Objeto (POO) é introduzido com a linguagem Smalltalk (**1983**)

Histórico de Orientação a Objetos (OO)

- FINS DOS ANOS **80** ⇒ Paradigma de Orientação a Objetos
 - abordagem poderosa e prática para o desenvolvimento de software
- Linguagens orientadas a objetos
 - Smalltalk (1972), Ada (1983), Eiffel (~1985)
 - Object Pascal (1986), Common Lisp (1986), C++ (1986)
 - Java (~1990), Python (~1990), Perl 5 (2005)

Linguagens orientadas a objetos

- "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](#).

Vantagens de OO

- 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;

Vantagens de OO

- 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;
- 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.

Vantagens de OO

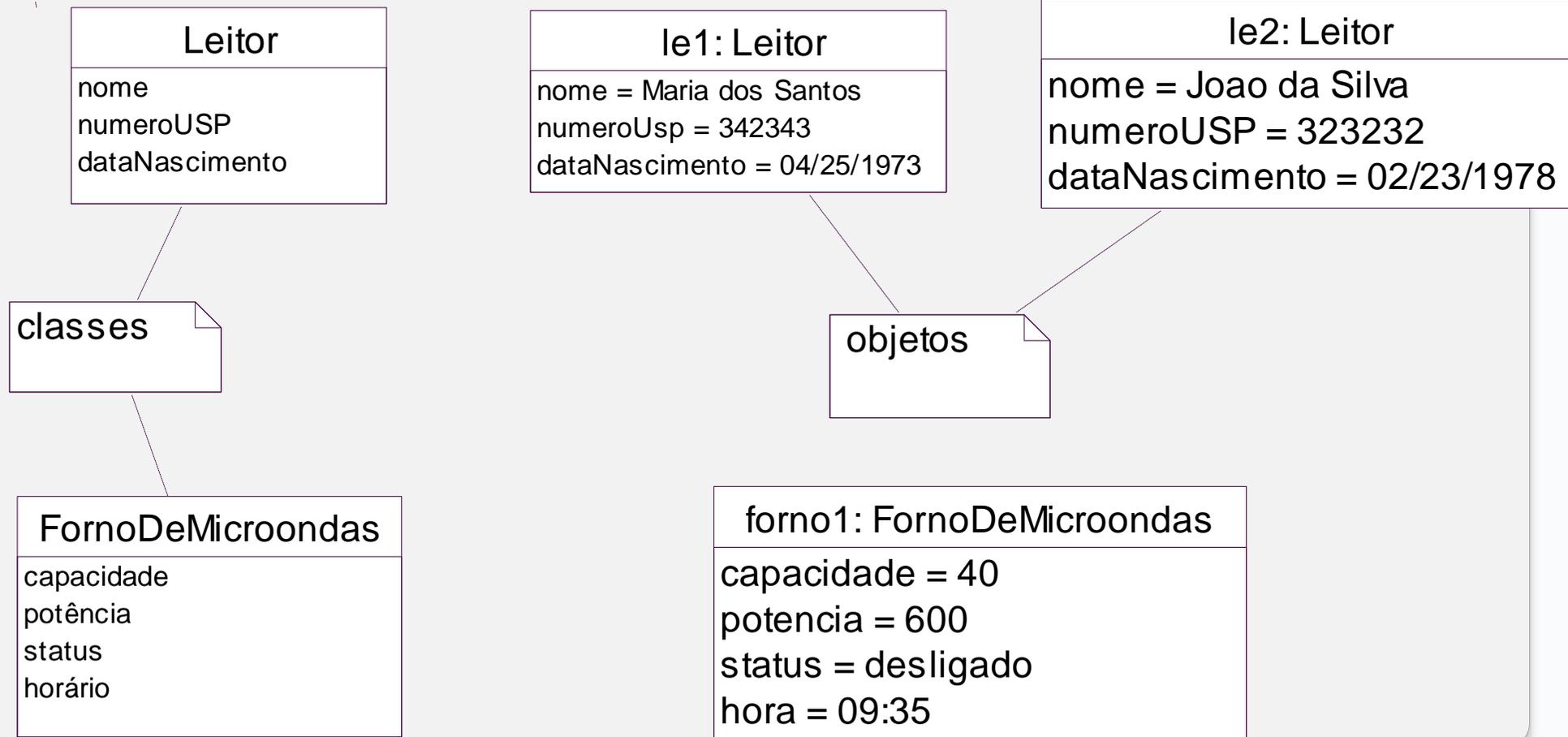
- maior dedicação à fase de análise, preocupando-se com a essência do sistema;
- mesma notação é utilizada desde a fase de análise até a implementação.

Frente a essas vantagens, a abordagem OO tem provado ser “popular” e eficaz.

Objetos

- Tudo em OO é OBJETO
 - Objeto, no mundo físico, é tipicamente um produtor e consumidor de itens de informação
- Definição (mundo do software)
 - *“Qualquer coisa, real ou abstrata, a respeito da qual armazenamos dados e métodos que os manipulam”*
Martin, Odell (1995)
 - Abstração de uma entidade do mundo real de modo que essa entidade possui várias características

Objetos e Classes



Classes

- Agrupamento de objetos similares.
- Todo objeto é uma instância de uma Classe.
- Os 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.

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

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

Métodos X Mensagem

mensagem

`le1.alterarNome('Rosa Olivera')`



le1: Leitor

nome = Maria dos Santos

numeroUsp = 342343

dataNascimento = 04/25/1973

método

método alterarNome(Char[30] novoNome)

Início

nome := novoNome;

Fim

Atributos e Métodos

Automóvel
proprietário marca placa ano
registrar transferir_Proprietário mudar_Placa



Atributos



Métodos

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

Abstração

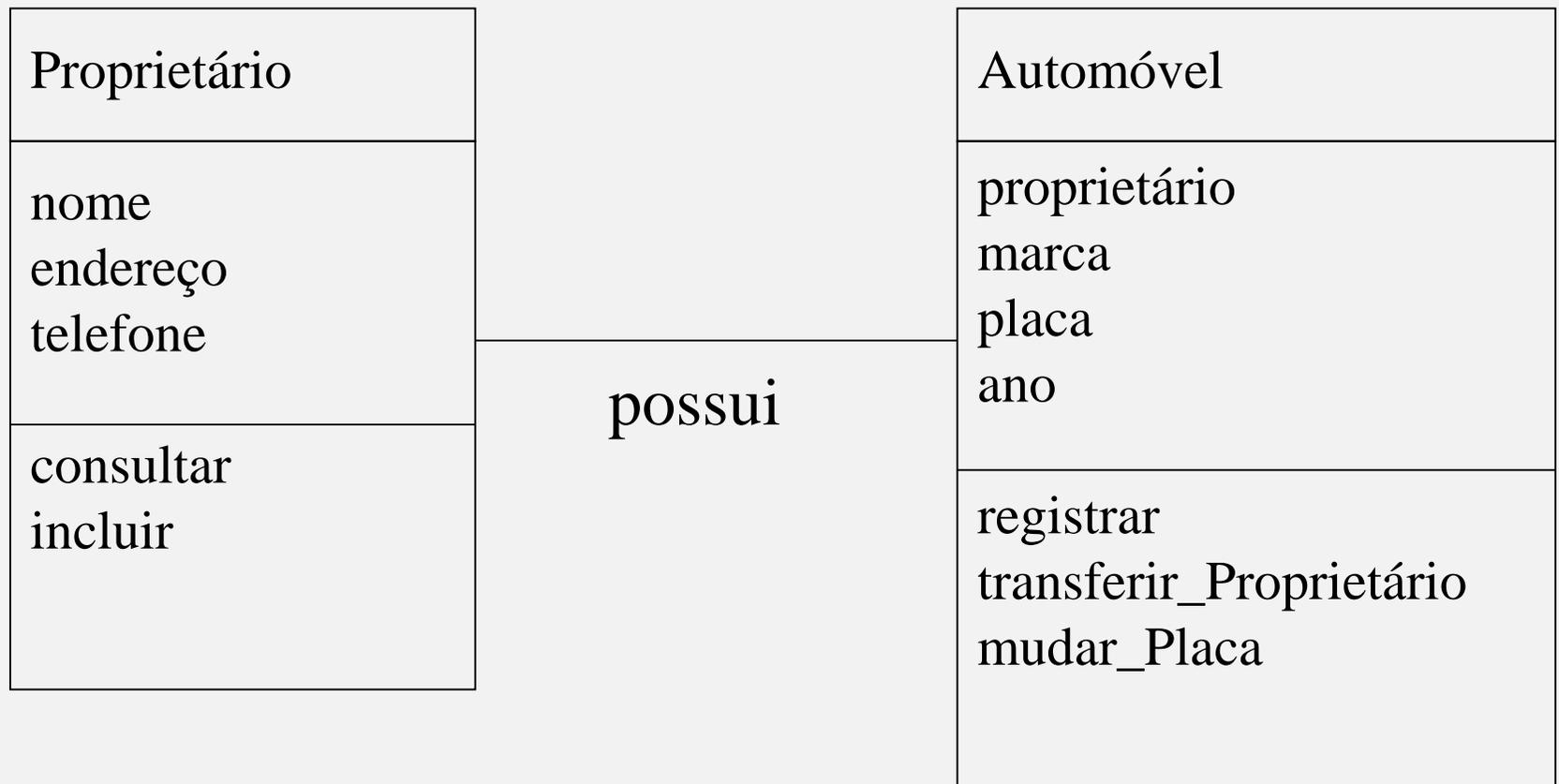


Encapsulamento

- 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 reuso

Conceitos Básicos

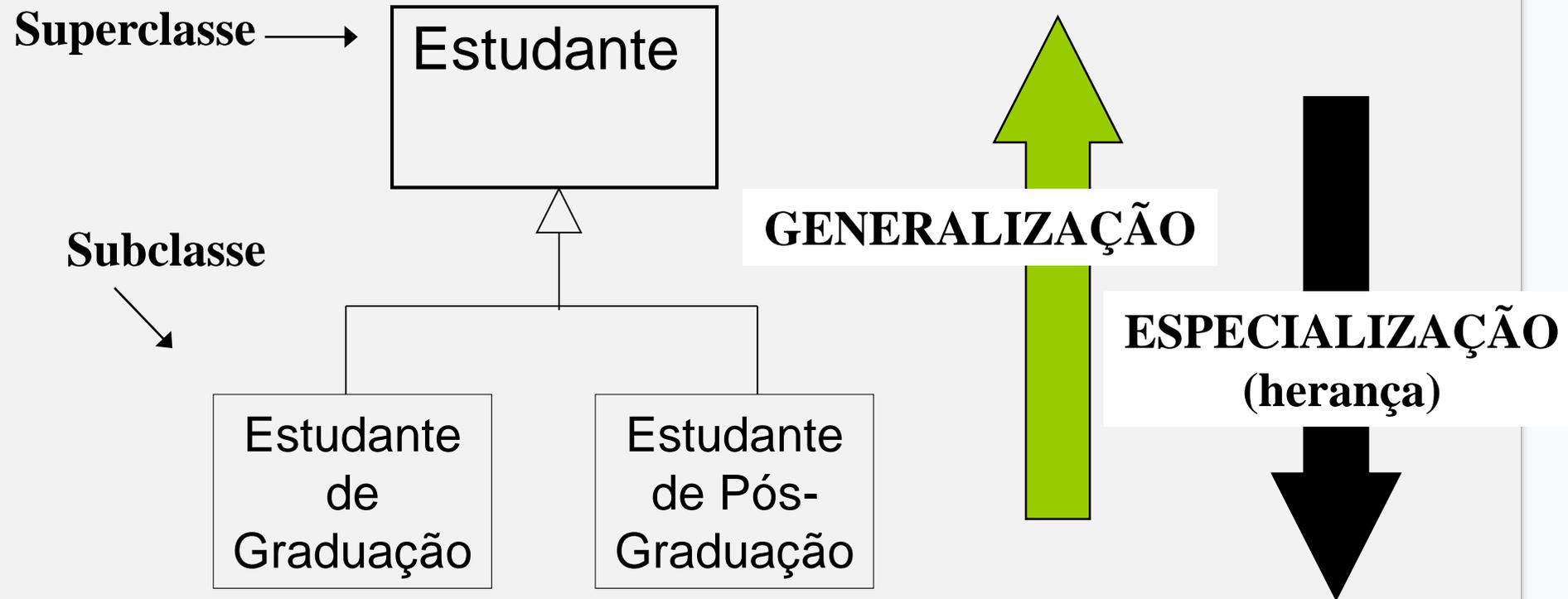
- Associações entre Classes



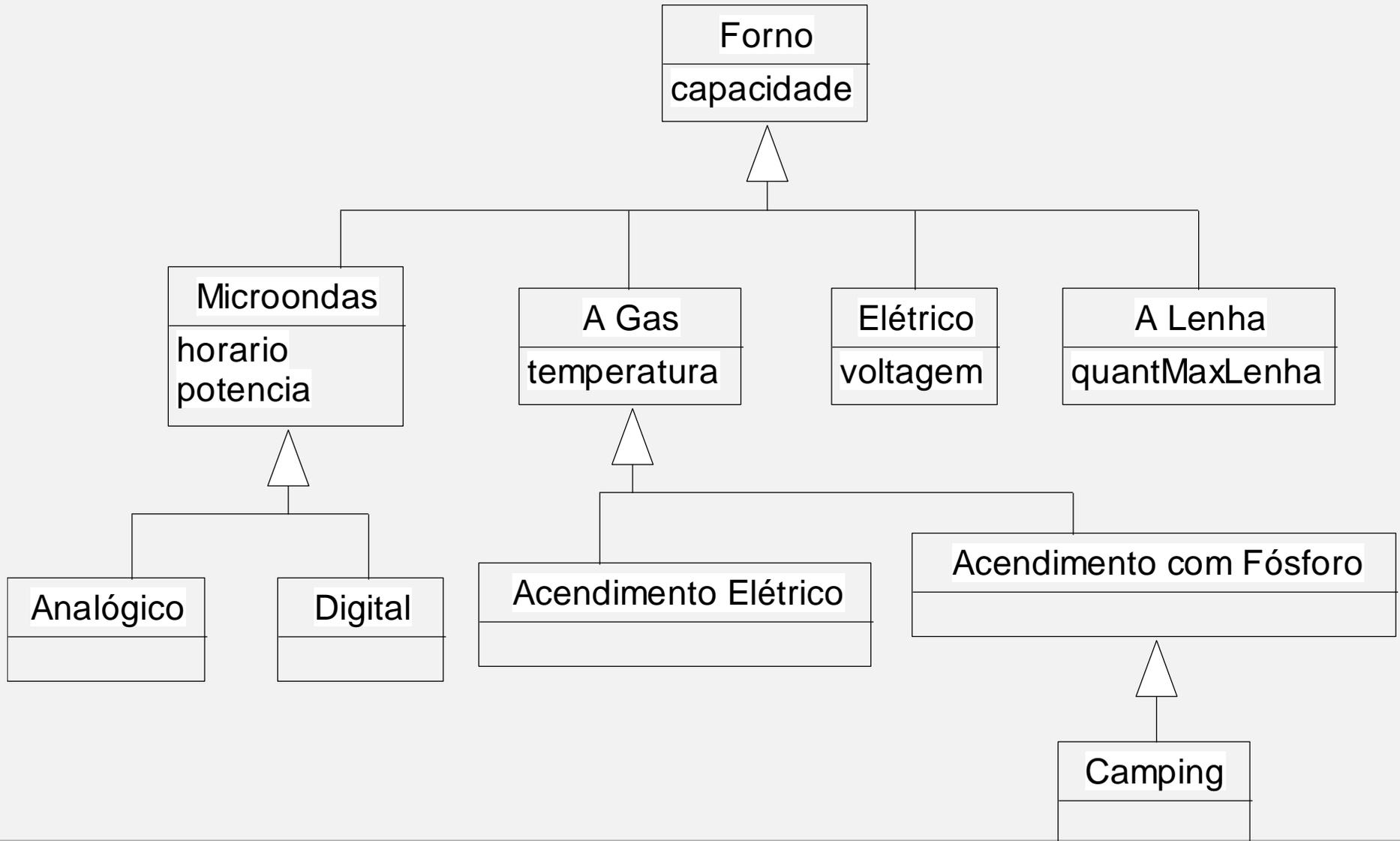
Herança

- 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.

Herança: Generalização/Especialização

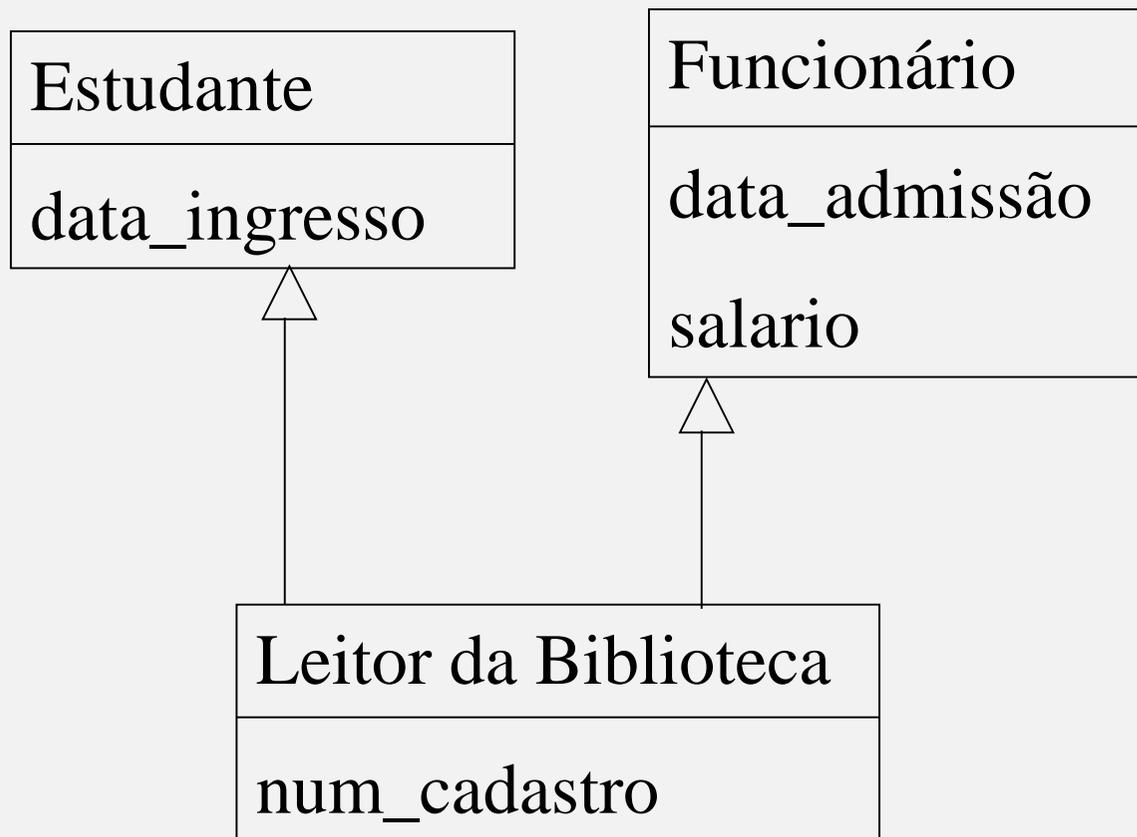


Herança



Herança Múltipla

Existe mais de uma superclasse, ou seja, uma classe é declarada como uma subclasse de uma ou mais superclasses.



Polimorfismo

- O Polimorfismo geralmente representa a qualidade ou estado de um objeto ser capaz de assumir diferentes formas.
- Mais especificamente, propriedade segundo o qual vários métodos podem existir com o mesmo nome.
 - Ao receber uma mensagem para efetuar uma Operação, é o objeto quem determina como a operação deve ser efetuada;
 - Permite a criação de várias classes com interfaces idênticas, porém objetos e implementações diferentes.

Polimorfismo

- Exemplos:
 - O operador “+” pode ser usado com inteiros, pontos-flutuantes ou *strings*.
 - A operação *mover* pode atuar diferentemente nas classes Janela e Peça de Xadrez.

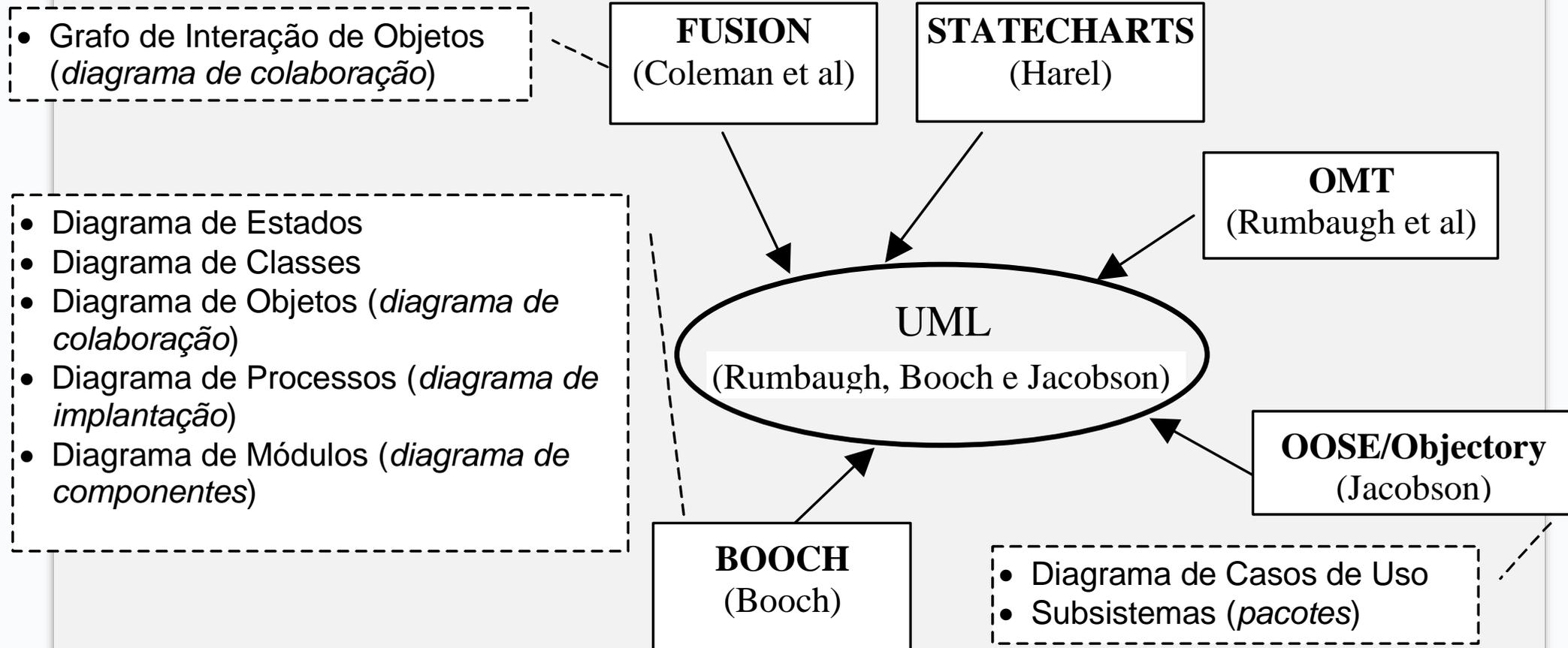
Introdução à UML

- UML (*Unified Modelling Language*)
- É uma linguagem para especificação, construção, visualização e documentação de sistemas.
- É uma evolução das linguagens para especificação de conceitos de *Booch*, OMT e OOSE e também de outros métodos de especificação de requisitos de software orientados a objetos ou não.

Histórico da UML

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

Introdução: UML



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.

Ferramentas de Apoio

- Exemplos:

- A família Rational Rose Enterprise (da *Rational Software Corporation* www.rational.com) que gera código em Smalltalk, PowerBuilder, C++, J++ e VB.
- **ArgoUML- free** <http://argouml.tigris.org/>
- www.objectsbydesign.com/tools/umltools_byCompany.html (lista de ferramentas que envolvem a UML), entre elas Jude (agora Astah) e Visual Paradigm
- StarUML: Livre
 - <http://staruml.sourceforge.net>

Diagramas da UML

- Diagramas de Casos de Uso
- Diagramas de Classe
- Diagramas de Comportamento
 - Diagrama de Estado
 - Diagrama de Atividade
 - Diagrama de Seqüência
 - Diagrama de Colaboração
- Diagramas de Implementação
 - Diagrama de Componente
 - Diagrama de Implantação (*Deployment*)

Material sobre UML

- <http://www.rational.com> (Rational)
- <http://www.omg.org> (*Object Management Group*)
- Page-Jones, M.; Fundamentos do desenho orientado a objeto com UML, Makron Books, 2001.
- Furlan, J. D.; Modelagem de Objetos Através da UML, Makron Books, 1998.
- Rumbaugh, J., Jacobson, I., Booch, G.; The Unified Modeling Language Reference Manual, Addison-Wesley, c1999.
- Conallen, J.; Building Web Applications with UML, Addison-Wesley, 1999.
- Fowler, M.; Scott, K.; UML Essencial, Bookman, 2000.